Electronic Colloquium on Computational Complexity, Report No. 62 (2023)

Conflict Checkable and Decodable Codes and Their Applications

Benny Applebaum* Eliran Kachlon*

May 2, 2023

Abstract

Let C be an error-correcting code over a large alphabet q of block length n, and assume that, a possibly corrupted, codeword c is distributively stored among n servers where the ith entry is being held by the ith server. Suppose that every pair of servers publicly announce whether the corresponding coordinates are "consistent" with some legal codeword or "conflicted". What type of information about c can be inferred from this consistency graph? Can we *check* whether errors occurred and if so, can we find the error locations and effectively *decode*? We initiate the study of *conflict-checkable* and *conflict-decodable* codes and prove the following main results:

(1) (Almost-MDS conflict-checkable codes:) For every distance $d \leq n$, there exists a code that supports conflict-based error-detection whose dimension k almost achieves the singleton bound, i.e., $k \geq n - d + 0.99$. Interestingly, the code is non-linear, and we give some evidence that suggests that this is inherent. Combinatorially, this yields an n-partite graph over $[q]^n$ that contains q^k cliques of size n whose pair-wise intersection is at most $n - d \leq k - 0.99$ vertices, generalizing a construction of Alon (Random Struct. Algorithms, '02) that achieves a similar result for the special case of triangles ($n = 3$).

(2) (Conflict Decodable Codes below half-distance:) For every distance $d \leq n$ there exists a linear code that supports conflict-based error-decoding up to half of the distance. The code's dimension k "half-meets" the singleton bound, i.e., $k = (n - d + 2)/2$, and we prove that this bound is tight for a natural class of such codes. The construction is based on symmetric bivariate polynomials and is rooted in the literature on verifiable secret sharing (Ben-Or, Goldwasser and Wigderson, STOC '88; Cramer, Damgård, and Maurer, EUROCRYPT '00).

(3) (Robust Conflict Decodable Codes:) We show that the above construction also satisfies a non-trivial notion of robust decoding/detection even when the number of errors is unbounded and up to $d/2$ of the servers are Byzantine and may lie about their conflicts. The resulting conflict-decoder runs in exponential time in this case, and we present an alternative construction that achieves quasipolynomial complexity at the expense of degrading the dimension to $k = (n - d + 3)/3$. Our construction is based on trilinear polynomials, and the algorithmic result follows by showing that the induced conflict graph is structured enough to allow efficient recovery of a maximal vertex cover.

As an application of the last result, we present the first polynomial-time statistical two-round Verifiable Secret Sharing (resp., three-round general MPC protocol) that remains secure in the presence of an active adversary that corrupts up to $t < n/3.001$ of the parties. We can

*Tel-Aviv University, Israel bennyap@post.tau.ac.il, elirn.chalon@gmail.com.

ISSN 1433-8092

upgrade the resiliency threshold to $n/3$, which is known to be optimal in this setting, at the expense of increasing the computational complexity to be quasipolynomial. Previous solutions (Applebaum, Kachlon, and Patra, TCC'20) suffered from an exponential-time complexity even when the adversary corrupts only $n/4$ of the parties.

Contents

1 Introduction

Error-correcting codes play an important role in theory and practive. They allow us to protect the integrity of data, both in communication and storage, and even the soundness of computation, in the context of interactive and non-interactive proof systems. In the classical setting, the decoder gets full access to a (possibly corrupted) codeword and its goal is to detect or correct errors. However, in the last 30 years a large amount of work was dedicated to codes that support decoding with *restricted access* to the codeword, with locally testable codes and locally decodable codes as the most notable examples (see [Gol10] and references therein).

Motivated by cryptographic applications, we present a new notion of decoding with restricted access to the codeword. For the sake of concreteness, consider the following (non-cryptographic) setting. Let $\mathcal{C} \subseteq [q]^n$ be a code, and let $\mathbf{w} \in [q]^n$ be a (possibly noisy) codeword that is distributed among n servers, where the ith server holds the ith entry $\mathbf{w}[i]$ of $\mathbf{w}$. The decoder has restricted access to the codeword $\mathbf{w}$, and it is only given access to the *conflict graph* G of $\mathbf{w}$. The conflict graph G of $\mathbf{w}$ is an undirected graph over n vertices, such that an edge (i, j) exists if and only if the ith and jth symbol of $\mathbf{c}$ are inconsistent with each other, i.e., if there is *no* codeword $\mathbf{c} \in \mathcal{C}$ that satisfies $\mathbf{c}[i] = \mathbf{w}[i]$ and $\mathbf{c}[j] = \mathbf{w}[j]$. The goal of the decoder is to check the validity of the codeword, and, if possible, to identify the error locations. This is challenging since the decoder sees only a tiny amount of information, about n^2 bits, whereas the entire codeword is of bit-length $n \log q$. In a typical setting where the number of servers is much smaller than the data-size, (e.g., $q = 2^{n^3}$) the number of bits that are available to the decoder may not even suffice for representing a *single* symbol of the alphabet.

Such a mechanism is useful in cases where a client wishes to read the codeword but has only an expensive line of communication to the servers, which in-turn, are connected with each other via a fast network. (Think of a server farm on the moon.) Instead of reading the entire codeword, the servers can compute the pair-wise consistencies in a single round of communication, and send the resulting graph (i.e., $O(n^2)$ bits) to the client. Based on a conflict-decoder, the client can then identify a set of uncorrupted locations, and by reading only the content of these servers, recover the information word. That is, standard decoding is *decomposed* into a conflict computation, which can be computed by the servers in a single round of interaction over point-to-point channels, conflict-decoding which is performed by the client, and actual decoding that requires only a minimal amount of data reads.[1]

The ability to check and decode given few bits of information is also very useful in cryptographic scenarios where some information about the codeword should be hidden. Indeed, the problems of conflict-decodability and conflict-checkability, (combined with various secrecy requirements), implicitly appear in the cryptographic literature about secure multiparty computation and verifiable secret sharing, starting with the classical works of [BGW88, CCD88, RB89].

In this paper, we initiate a systematic study of conflict-based decoding from a purely coding-theoretic perspective. We consider different tasks such as error detection and error correction under different noise models, and present definitions, constructions, and lower bounds, in an attempt to understand how conflict-decoding affects the possible trade-offs between the rate and the distance (typically over large alphabets). We will also discuss some applications of this new

[1]Similarly, conflict-checking provides a communication efficient way to ensure that the codeword is non-noisy, this can be used repeatedly to ensure validity. When a noise is detected, a more expensive correction procedure can be applied.

framework, and compare it to existing notions such as locally-testable codes. We continue with a formal presentation of our results, starting with the most basic notion of *conflict checkable codes*.

1.1 Conflict Checkable Codes

In conflict checkable codes, checking whether a vector $\mathbf{c}$ is a codeword is done by only inspecting its conflict graph: $\mathbf{c}$ is a codeword if and only if its conflict graph is empty (i.e., it contains no edges). This is formalized in the following definition.

Definition 1.1 (Conflict functions and graphs). *For a code $\mathcal{C} \subset [q]^n$ and every indices $i < j \in [n]$, we define the (i,j)-th* conflict function $G_{i,j} : [q] \times [q] \to \{0,1\}$ *of $\mathcal{C}$ to be the function that, given $\sigma, \tau \in [q]$ outputs 0 if and only if there exists $\mathbf{c} \in \mathcal{C}$ such that $\mathbf{c}[i] = \sigma$ and $\mathbf{c}[j] = \tau$. The* conflict functions *of $\mathcal{C}$ is defined to be $G = (G_{i,j})_{1 \le i < j \le n}$.*

For a vector $\mathbf{c} \in [q]^n$ (not necessarily a codeword), we define the conflict graph K *of $\mathbf{c}$ to be the (undirected) graph on n vertices $1, \ldots, n$, where there is an edge (i,j) if and only if $G_{i,j}(\mathbf{c}[i], \mathbf{c}[j]) = 1$. We say that $\mathbf{c}[i]$ is* conflicted *with $\mathbf{c}[j]$ when an the edge (i,j) exists, and say that $\mathbf{c}[i]$ is* consistent *with $\mathbf{c}[j]$ otherwise. The code $\mathcal{C}$ is* conflict checkable *if for every $\mathbf{c} \in [q]^n$ it holds that $\mathbf{c} \in \mathcal{C}$ if and only if $G_{i,j}(\mathbf{c}[i], \mathbf{c}[j]) = 0$ for every $1 \le i < j \le n$, i.e., the conflict graph of $\mathbf{c}$ is empty.*

Simple examples and non-examples. We note that the repetition code $\mathcal{C} = \{(\sigma, \ldots, \sigma) \mid \sigma \in [q]\}$ is an $(n, k = 1, d = n)_q$ conflict checkable code and that the trivial code $\mathcal{C} = [q]^n$ is an $(n, k = n, d = 1)_q$ conflict checkable code. These codes satisfy the Singleton bound $k \le n - d + 1$ with equality, and therefore they are *maximum distance separable* (MDS) codes. It is not hard to see that these are the only examples of conflict-checkable codes that meet the Singleton bound. Indeed, whenever $k \ge 2$, every two entries (i,j) of an MDS code are independent, i.e., for every $\sigma, \tau \in [q]$ there exists a codeword $\mathbf{c}$ with $\mathbf{c}[i] = \sigma$ and $\mathbf{c}[j] = \tau$, so the functions $G_{i,j}$ always return 0, and the code must contain the set of all the possible vectors. Similarly, a linear code $\mathcal{C}$ whose *dual code* $\mathcal{C}^\perp$ has distance $d^\perp \ge 3$ *cannot* be conflict checkable (unless it is the trivial code), since every $d^\perp - 1 \ge 2$ rows of the generating matrix of $\mathcal{C}$ are linearly independent. It follows that some of the most well-studied codes, such as Reed-Solomon, Reed-Muller, Hadamard and random linear codes, are *not* conflict checkable since they have non-trivial dual distance.

Comparison to 2-LTCs. From a definitional perspective, an access to conflicts is very different from local access (as in locally testable codes): the former consists of $O(n^2)$ *bits* of information that globally depend on the codeword whereas the latter information is local and consists of few code symbols (that may potentially contain many bits). The use of global information also allows us to use deterministic checkers and to require perfect correctness and soundness even for words that are very close to the code – properties that cannot be achieved by locally-testable code. Nevertheless, it is not hard to see that 2-query locally-testable codes (2-LTC) are also conflict checkable, since every non-codeword violates a positive fraction of the pairwise consistency checks.[2] Consequently, one can use the the recent breakthrough LTC construction of Dinur et al. [DEL+22] to obtain conflict-checkable codes with constant rate, and constant relative-distance over constant-size

[2] Technically, we need a variant of LTC with 1-sided errors in which every non-codeword is rejected with positive probability. This is implied, for example, by *strongly* local testablity as defined in [Gol10, Section 2.3.1]. We also mention that constant-query LTC can be turned into a 2-LTC via standard techniques (e.g., [Din07]): Pack every tuple that is queried by a test into a "super-symbol" and let the 2-query test verify the consistency of "super-symbols".

alphabet. We note that this may be an overkill, as 2-testability seems like a significantly stronger notion than conflict-checkability; In the former case, vectors that are far from the code should violate a *large* fraction of the (possibly weighted) pairwise consistency checks, whereas in the latter case such vectors are only required to violate *some* conflicts. In addition, while small alphabet is typically an important feature of LTCs, we will typically be interested in the large-alphabet regime.

The above examples suggest that conflict-checkability requires some redundancy among *pairs* of entries but possibly less redundancy than is needed for LTCs. Of course, the interesting question is how much redundancy is needed. Somewhat surprisingly, we show that just a "tiny" amount of redundancy is needed, and it is possible to obtain conflict checkable codes that almost achieve the Singleton bound (beating the existing upper-bounds for LTCs). Before presenting our construction, it will be instructive to adopt an alternative view of conflict checkable codes.

A combinatorial view. Definition 1.1 provides an algorithmic view of conflict checkable codes: Given the conflict graph K of a vector $\mathbf{c}$ one can decide whether $\mathbf{c}$ is a codeword. Taking a more combinatorial view, we can identify the conflict functions $G = (G_{i,j})_{1 \le i < j \le n}$ of the code with a complete n-partite undirected graph $\mathcal{G}$ over q-size sets of vertices, $V_1, \ldots, V_n$, whose edges are labeled by 1 (*conflicted*) or 0 (*consistent*) according to the conflict functions. That is, the edge from the σ-th vertex in V_i to the τ-th vertex in V_j is labeled by $G_{i,j}(\sigma, \tau)$. By definition, any codeword $\mathbf{c} \in \mathcal{C}$ corresponds to an n-size clique of *consistent* edges. It is not hard to see that the converse also holds.

Claim 1.2. *A code $\mathcal{C} \subset [q]^n$ is conflict checkable if and only if there exists an n-partite graph $H = (V_1, \ldots, V_n, E)$ with the following properties:*

1. *For every $i \in [n]$, the i-th part V_i consists of q vertices denoted by $(i, 1), \ldots, (i, q)$.*
2. *The code $\mathcal{C}$ consists of all vectors $\mathbf{c} \in [q]^n$ for which $(1, \mathbf{c}[1]), \ldots, (n, \mathbf{c}[n])$ forms a clique in H.*

Proof. If $\mathcal{C}$ is conflict checkable, take H to be the restriction of the n-partite graph $\mathcal{G}$ (defined by the conflict functions) to the consistent edges, and observe that the graph satisfies the required properties. For the other direction, assume that the codewords of a code $\mathcal{C}$ correspond to cliques in some n-partite graph H with q vertices on each side. Observe that the conflict functions $G = (G_{i,j})$ of $\mathcal{C}$ satisfy $G_{i,j}(\sigma, \tau) = 0$ only if $((i, \sigma), (j, \tau)) \in E$. Therefore every $\mathbf{c} \in [q]^n$ is a codeword if and only if it induces an empty conflict graph, so $\mathcal{C}$ is conflict checkable. $\square$

Thus the design of conflict checkable codes with a good rate and a good distance boils down to packing as many cliques as possible in an n-partite graph while making sure that each pair of cliques intersects in a small number t of vertices. Indeed, if each part of the graph contains q nodes and the number of cliques is q^k, this yields an $(n, k, d)_q$ code with distance of $d = n - t$.

1.1.1 Almost-Optimal Conflict Checkable Codes

In Section 2 we provide a (non-explicit) construction of a conflict checkable code that almost satisfies the Singleton bound.

Theorem 1.3. *For every integers $n \ge 3$ and $2 \le d \le n - 1$, and every $\epsilon > 0$, there exists an integer $q = q(n, d, \epsilon)$ for which there exists an $(n, k, d)_q$ conflict checkable code for $k \ge n - d + 1 - \epsilon$.*

We emphasize that our code is not linear, and therefore the dimension k is not necessarily an integer. From a combinatorial point-of-view, we prove that for every $n \geq 3$ and $2 \leq d \leq n-1$, and every $\epsilon > 0$, there exists an integer $q = q(n, d, \epsilon)$ for which there exists an n-partite undirected graph over q-size sets of vertices $V_1, \ldots, V_n$ that contains at least q^k cliques, where each pair of cliques intersects in at most $t = n - d$ vertices. As we've mentioned, for a distance $2 \leq d \leq n-1$ the dimension must satisfy $k < n - d + 1$, and therefore the bound on k is almost optimal. We also mention that the size of the alphabet q is relatively large, approximately $2^{(n/\epsilon)^2}$.

Our construction. Alon [Alo02, Section 3], following the construction of Ruzsa and Szemerédi [RS78], provided a construction of a 3-partite graph, with q vertices on each side, that contains at least $q^{2-\epsilon}$ edge-disjoint triangles (that is, every pair of triangles intersects in at most a single vertex). The construction employs a result of Behrend [Beh46] about the existence of dense subsets of integers in $[q]$ containing no 3-term arithmetic progressions. Our construction can be seen as an extension and generalization of Alon's construction to *cliques* in *n-partite graphs*, where every pair of cliques is allowed to intersect in *at most t vertices*.

At a high level, we first construct a large set $\mathcal{F}$ of size $q^{n-d+1-\epsilon}$ containing univariate degree-t polynomials (over the integers), for $t = n - d$, that satisfy some special property (*) that will be discussed shortly. We begin with the standard evaluation-based code $\mathcal{C}'$ in which every polynomial $f \in \mathcal{F}$ is mapped to the codeword $(f(i))_{i \in [n]}$. We then consider the n-partite graph H that is induced by the code, i.e., for every polynomial $f \in \mathcal{F}$ we place an n-size clique over the n vertices $(1, f(1)), (2, f(2)), \ldots, (n, f(n))$. Finally, we define our code $\mathcal{C}$ based on H, i.e., by taking all the cliques in H. By design, there are at least $|\mathcal{F}| = q^{n-d+1-\epsilon}$ such cliques since every polynomial $f \in \mathcal{F}$ induces a unique clique. (In fact, any pair of such cliques intersects in at most t vertices since the polynomials are of degree t). However, the graph H may contain some additional cliques of size n that are *not* induced by polynomials in $\mathcal{F}$, and the main challenge is to show that *any* pair of cliques of size n intersects in at most t vertices.

In order to extend the argument for general cliques, it suffices to prove that for every clique $R = (i, \sigma_i)_{i \in [n]}$ of size n in H there exists a degree-t polynomial $G_R(x)$ that is consistent with R, i.e., $G_R(i) = \sigma_i$ for all $i \in [n]$. At a high level, we use the special property (*), to show that *every clique R' of size $t+2$* in H is consistent with some *degree-t polynomial $G_{R'}$*. This means that, for every $i \in [n]$, there exists a degree-t polynomial G_i that is consistent with the $(t+2)$-size sub-clique R_i of R that contains the first $t+1$ elements of R plus the ith element. The polynomials $G_1, \ldots, G_n$ agree on the first $t+1$ inputs and so they are all equal to a single polynomial G_R which is consistent with the n-clique R, as required.

Finally, let us provide some details regarding the structure of the set $\mathcal{F}$. At a high level, the set consists of degree-t univariate polynomials $f(x) = a_0 + a_1 x + \ldots + a_t x^t$, where we think of each a_i as a vector $\mathbf{v}_i$ that corresponds to its unique representation in base b, for some appropriately chosen integer b. We put two (non-linear) limitations on these vectors: (1) each entry of $\mathbf{v}_i$ is relatively small, and accordingly (a bounded number of) arithmetic operations over the coefficients are translated to operations over the vectors; and (2) For every index i, we fix the norm of the vectors $\mathbf{v}_i$ in *all* polynomials to be some value B_i, and for every pair of indices i, j we fix the inner product $\mathbf{v}_i \cdot \mathbf{v}_j$ in *all* polynomials to be some value $B_{i,j}$. Most of the technical work is dedicated to the proof that this additional level of redundancy allows us to extract a degree-t polynomial for every $t+2$-size clique. An averaging argument further shows and that there exists a choice of $b, (B_i, B_{i,j})_{i,j}$ for which the set $\mathcal{F}$ can be sufficiently large. See Section 2 for full details.

1.2 Conflict Decodable Codes

In conflict checkable codes we used the conflict graph to check whether a word belongs to the code. We extend this definition to *conflict-decodability* as follows.

Definition 1.4 (Conflict decodable codes). *A code $\mathcal{C} \subseteq [q]^n$ is a t-*conflict decodable code *if there exists an algorithm F so that for every word $\mathbf{w} \in [q]^n$ that is at most t-far from a codeword $\mathbf{c} \in \mathcal{C}$ the following holds. The algorithm F, given the conflict graph of $\mathbf{w}$, returns a set of indices $I \subseteq [n]$ such that $\mathbf{c}$ is the only codeword that satisfies $\mathbf{c}[i] = \mathbf{w}[i]$ for all $i \in I$.*

Remark 1.5 (Reducing error correction to recovery from erasures). *We observe that conflict decodable codes allow us to reduce correction from t errors to recovery from erasures. Indeed, given a word $\mathbf{w}$ that is at most t-far from a codeword $\mathbf{c} \in \mathcal{C}$, we execute F on the conflict graph of $\mathbf{w}$ to obtain the set of indices I, and for every $i \in [n] \setminus I$ we set the ith entry of $\mathbf{w}$ to an erasure, i.e., $\mathbf{w}[i] = \perp$. Finally, we execute the recovery from erasures algorithm on (the modified) $\mathbf{w}$ to obtain $\mathbf{c}$. Since $\mathbf{c} \in \mathcal{C}$ is the only codeword that satisfies $\mathbf{c}[i] = \mathbf{w}[i]$ for all $i \in I$, we are guaranteed to obtain the codeword $\mathbf{c}$. This implies that we can recover from t errors, so the best one can hope for is $t \leq \lfloor (d-1)/2 \rfloor$, where d is the distance of $\mathcal{C}$. By default, we will set t to $\lfloor (d-1)/2 \rfloor$.*

A sufficient condition for conflict-decodability. It turns out that any conflict-checkable code that satisfies the following natural local-to-global consistency property is also conflict-decodable.

Lemma 1.6 (checkability and Local-to-global consistency $\Rightarrow$ decodability). *An $(n, k, d)_q$ conflict checkable code $\mathcal{C}$ provides* local-to-global consistency *if for every set $I \subseteq [n]$ of at least $n - d + 1$ indices and every symbols $(\sigma_i)_{i \in I}$, if $G_{i,j}(\sigma_i, \sigma_j) = 0$ for every $i, j \in I$ such that $i < j$, then there exists a codeword $\mathbf{c} \in \mathcal{C}$ such that $\mathbf{c}[i] = \sigma_i$ for every $i \in I$.*[3]

Any $(n, k, d)_q$ conflict checkable code $\mathcal{C}$ that satisfies local-to-global consistency is also t-conflict decodable with $t = \lfloor (d-1)/2 \rfloor$. Moreover, the conflict-decoding algorithm F is efficiently computable.

By Remark 1.5, it follows if the code $\mathcal{C}$ is linear (and so one can efficiently compute the conflict graph and can efficiently decode under erasures), then it also has an efficient (standard) error correcting algorithm that corrects up to $\lfloor (d-1)/2 \rfloor$ errors.

Proof of Lemma 1.6. Let $\mathbf{w}$ be a noisy codeword with distance at most $\lfloor (d-1)/2 \rfloor$ from a codeword $\mathbf{c} \in \mathcal{C}$, and let K be the inconsistency graph of $\mathbf{w}$. Given an input K, the conflict decoder F finds a 2-approximation vertex cover E in K and outputs $I := [n] \setminus E$. The vertex cover is chosen by running the classic efficient greedy algorithm that repeatedly picks an edge, adds its two vertices to the vertex cover and removes them from the graph (see, e.g., [CLRS09, Section 35.1]).

The analysis is straightforward. Every edge in K is incident on at least one vertex that corresponds to a noisy entry, so the noisy entries form a vertex cover of size at most $\lfloor (d-1)/2 \rfloor$. Therefore, the size of E is at most $d - 1$, and at most $|E|/2 \leq (d-1)/2$ of the vertices in E correspond to non-noisy entries in $\mathbf{w}$. Recall that the total number of non-noisy entries is at least $n - \lfloor (d-1)/2 \rfloor$, and so the number of non-noisy entries in I is at least $(n - \lfloor (d-1)/2 \rfloor) - (d-1)/2 \geq n - d + 1$. Since all entries in I are pairwise consistent, the local-to-global consistency implies that there exists a unique codeword that agrees with all entries in I, and since at least $n - d + 1$ entries in I are non-noisy, this codeword has to be $\mathbf{c}$. This concludes the proof of the lemma. $\square$

[3]Since the distance is d, the codeword $\mathbf{c}$ has to be *unique*.

We do not know whether the code from Theorem 1.3 satisfies local-to-global consistency. We construct conflict decodable codes by showing that codes that are based on bivariate polynomials satisfy local-to-global consistency.

1.2.1 Conflict Decodable Codes from Symmetric Bivariate Polynomials

A natural way of obtaining dependency among pairs of entries of a codeword is by considering the restriction of *multivariate polynomials* into linear subspaces. Indeed, this method appears in the literature of probabilistic checkable proofs (see, e.g., [FGL+91, AS98, ALM+98]) and locally testable codes (see, e.g., [RS92, RS97, BDN16]), as well as in the cryptographic literature, in the context of verifiable secret sharing (see, e.g., [BGW88, CDM00, KKK09]). In fact, locally testable codes based on *tensoring* (see, e.g., [BSS04, Mei08]) can be seen as generalizations of these ideas.

We show that this approach also applies for the construction of conflict decodable codes, by considering the following code, that appears implicitly in [CDM00, KKK09]. Let $\mathbb{F}_p$ be a finite field of size $p \geq n$, let $1, \ldots, n$ be n distinct non-zero field elements, and let $1 \leq \ell < n$ be an integer. Consider the following code,

$$\mathcal{C}_{\mathsf{bivariate}} = \left\{ (F(x,1), \ldots, F(x,n)) \,\middle|\, \begin{array}{c} F \text{ is a symmetric bivariate polynomial} \\ \text{of degree at most } \ell \text{ in each variable} \end{array} \right\},$$

where every codeword $\mathbf{c} \in \mathcal{C}_{\mathsf{bivariate}}$ corresponds to some symmetric bivariate polynomial $F(x,y)$ of degree at most ℓ in each variable, and the i-th entry of $\mathbf{c}$ is the degree-ℓ *univariate* polynomial $\mathbf{c}[i] = F(x,i)$ that is obtained from $F(x,y)$ via the substitution $y = i$. Such univariate polynomials can be naturally represented as vectors in $\mathbb{F}_p^{\ell+1}$. In Section 4.3 we prove the following theorem.

Theorem 1.7. *The code $\mathcal{C}_{\mathsf{bivariate}}$ is a linear*[4] $[n, k = (\ell+2)/2, d = n - \ell]_q$ *conflict checkable code that satisfies local-to-global consistency, with alphabet $q = p^{\ell+1}$.*

Therefore, by Lemma 1.6 the code $\mathcal{C}_{\mathsf{bivariate}}$ is a t-conflict decodable code for $t = \lfloor (n-\ell-1)/2 \rfloor$. In Section 4.3 we generalize this construction and show how to derive conflict-decodable codes by restricting a symmetric multilinear map according to the generating matrix of an arbitrary linear MDS code. Our framework generalizes and extends the secret-sharing construction of [PC12, Section 3.2.3] which, in turn, is based on [CDM00].

Comparison-based codes. The code $\mathcal{C}_{\mathsf{bivariate}}$ satisfies a special property: if $f_i(x)$ is the ith entry, and $f_j(x)$ is the jth entry, then the ith entry is consistent with the jth entry if and only if $f_i(j) = f_j(i)$. In other words, in order to compute the conflict function $G_{i,j}$ the servers i and j just need to apply an equality-check between a pair of locally-computable values. This feature, denoted *comparison-based conflicts*, simplifies the computation of conflicts, and will play an important role later when presenting the cryptographic applications. It is not hard to show that every linear conflict-decodable code supports comparison-based conflicts (see Section 4.2). The use of comparison-based conflicts, induces additional redundancy as, at least intuitively, each bit of information appears in 2 copies. By using tools from information theory, we formalize this intuition

[4] In this paper we refer to a code as *linear* if given an information word $\mathbf{w} \in \mathbb{F}^{k'}$, the i-th entry of the codeword can be obtained via a linear transformation. This is a generalization of the standard notion of linear codes, and therefore the dimension is not necessarily integral. See Section 4.2 for more details.

and prove in Section 4.1 that the combination of comparison-based conflicts and local-to-global consistency degrades the achievable rate by a factor of 2, compared to MDS codes.

Theorem 1.8. *For every $(n, k, d)_q$ comparison-based conflict checkable code with $1 < d < n$ that satisfies local-to-global consistency, it holds that $k \leq \frac{n-d+2}{2}$. In particular, the bound holds for any linear conflict checkable code that satisfies local-to-global consistency.*

Observe that our code $\mathcal{C}_{\mathsf{bivariate}}$ satisfies the theorem's conditions and "half-meets" the Singleton bound, i.e., it satisfies $k = (n - d + 2)/2$, and so, by Theorem 1.8, it achieves an optimal rate. Recall that our conflict-checkable code from Theorem 1.3 is an almost-MDS code and so it bypass the above bound. Indeed, the code is not comparison-based and is not known to achieve local-to-global consistency. We conjecture that the former property is the important one and that the bound from Theorem 1.8 can be bypassed by some conflict checkable code with local-to-global consistency. Observe that this conjecture holds for the special case of $d = n - 1$ since, in this case, the conflict checkable code from Theorem 1.3 trivially satisfies local-to-global consistency (any pair of consistent entries must be consistent with a *unique* codeword).

1.3 Robust Conflict Decodable Codes

Robust conflict decodable codes extend conflict decodable codes in two orthogonal ways: They provide some guarantees even when the noisy codeword is *far from the code* and even when the *conflict graph* is corrupted. Roughly, the first case corresponds to a scenario where the *writer* who stored the information on the servers was malicious and the second one deals with the case where some of the servers are malicious. The code should handle these two cases simultaneously, i.e., cope with a malicious coalition that includes a bad writer and up to t servers. Details follow.

Consider the scenario where a writer distributes some information $\mathbf{x} \in [q]^n$ among n remote servers, where the ith server holds $\mathbf{x}[i]$. Supposedly, the information is coherent, i.e., it corresponds to a codeword $\mathbf{c} \in \mathcal{C}$ where the ith server holds the ith entry $\mathbf{x}[i] = \mathbf{c}[i]$. Next, an honest reader wishes to read the information, and to save bandwidth, she first asks the servers to pairwise compare their data and then publish the conflict graph. The reader inspects the graph and should apply some form of decoding that should be robust even in the setting where the writer and a subset $\mathsf{B} \subseteq [n]$ of at most t servers is corrupted by an adversary. This means that the stored vector $\mathbf{x}$ may be far from the code, and that instead of seeing the conflict graph $\mathsf{K}(\mathbf{x})$ of $\mathbf{x}$, the reader only sees a modified version in which the adversary can fully add/remove edges that are incident to B. (Since the servers in B may lie about their conflicts.)

To cope with this situation, we relax the output of the decoder and, instead of asking her to output a single set I of non-noisy entries, she is allowed to output a list of vertex subsets, $\mathcal{L}$, where we think of each set $E \in \mathcal{L}$ as an "explanation" or a "guess" for the set of *corrupt* servers. Accordingly, E is a subset of $[n]$ of size at most t. We make the following requirements: (1) (Validity) Every pair of honest servers *outside the explanation* has to be consistent. (2) (Unique decoding for honest writer) If the writer is honest and wrote a codeword $\mathbf{c}$, then, no matter which explanation E is chosen from the list, the content of the honest servers outside E uniquely determines $\mathbf{c}$. Specifically, we can access the content of the servers outside E, and, assuming that bad servers can be identified via such an access (e.g., by some cryptographic mechanism or by inspecting their internal state), we can recover the stored codeword $\mathbf{c}$. (3) If the writer is malicious, then for every explanation in the list E, the honest parties outside the explanation uniquely define some codeword $\mathbf{c}' \in \mathcal{C}$ that may vary across different explanations. This effectively means that even a malicious writer is forced to write some codeword. (We will later discuss a stronger variant that guarantees that all explanations correspond to the same codeword.) Finally, it will be useful to strengthen Requirement (2) for an honest writer, and additionally require the existence of at least one explanation in the list containing only corrupt parties. Intuitively, this property allows us to support multiple write operations that were either employed by the same writer or by different writers. Indeed, if the writers are all honest, we can find a common t-size subset E that covers, in each list of explanations $\mathcal{L}_i$, at least one explanation $E_i \in \mathcal{L}_i$, and so we can access the servers in E and properly decode all the codewords.

We continue with a formal definition of robust conflict deocdable codes.

Definition 1.9 (Robust conflict decodable codes)**.** *Let $\mathcal{C} \subseteq [q]^n$ be a code whose conflict functions are $G = (G_{i,j})_{1 \leq i < j \leq n}$. For a vector $\mathbf{x} \in [q]^n$ (not necessarily a codeword) and for a set $\mathsf{B} \subseteq [n]$, we say that a graph K is* B-corrupt with respect to $\mathbf{x}$, *if K can be obtained from the conflict graph $\mathsf{K}(\mathbf{x})$ of $\mathbf{x}$ by adding and removing only edges that are incident on at least one vertex of B.*

*For an integer $0 \leq t \leq n$, we say that $\mathcal{C}$ is a t-*robust conflict decodable code *if there exists a function $\mathcal{E}$, called the* conflict-decoder function*, such that for every vector $\mathbf{x} \in [q]^n$, every set $\mathsf{B} \subseteq [n]$ of size at most t, and every graph K that is B-corrupt with respect to $\mathbf{x}$, the conflict-decoder function $\mathcal{E}$ takes the graph K and returns a (possibly empty)* explanation-list *$\mathcal{L}$ where every $E \in \mathcal{L}$ is a subset of $[n]$ of size at most t and the following holds.*

- (Validity of explanations) *For every explanation E in $\mathcal{L}$, and every pair i, j in $\mathsf{H} := [n] \setminus \mathsf{B}$ for which $G_{i,j}(\mathbf{x}[i], \mathbf{x}[j]) = 1$, either $i \in E$ or $j \in E$ (or both). That is, every explanation forms a vertex cover of the graph $\mathsf{K}[\mathsf{H}]$ that is obtained by restricting K the set of the honest locations.*

- (Good inputs) *If there exists a codeword $\mathbf{c} \in \mathcal{C}$ such that $\mathbf{x}[i] = \mathbf{c}[i]$ for every $i \in \mathsf{H}$, then (1) there exists an explanation E that is a subset of B (in particular $\mathcal{L}$ is not empty), and (2) for every explanation E, the codeword $\mathbf{c}$ is the only codeword that satisfies $\mathbf{x}[i] = \mathbf{c}[i]$ for all $i \in \mathsf{H} \setminus E$.*

- (Bad inputs) *If there is no codeword $\mathbf{c} \in \mathcal{C}$ such that $\mathbf{x}[i] = \mathbf{c}[i]$ for every $i \in \mathsf{H}$, then either $\mathcal{L}$ is empty, or for every explanation E there exists a unique codeword $\mathbf{c} \in \mathcal{C}$ such that $\mathbf{x}[i] = \mathbf{c}[i]$ for all $i \in \mathsf{H} \setminus E$.*

Note that the decoder is allowed to output an empty list of explanations if he "catches" a cheating writer.

On the guarantees for bad inputs. In Definition 1.9, when the inputs of the honest servers are bad (i.e., they are inconsistent with every codeword), different explanations can define different codewords. One could suggest a stronger definition, where there exists a codeword $\mathbf{c}$, that is consistent with every explanation:

> *(Strong guarantees for bad inputs)* If there is no codeword $\mathbf{c} \in \mathcal{C}$ such that $\mathbf{x}[i] = \mathbf{c}[i]$ for every $i \in \mathsf{H}$, then either $\mathcal{L}$ is empty, *or there exists a unique codeword $\mathbf{c} \in \mathcal{C}$ such that for every explanation E, only the codewrod $\mathbf{c}$ satisfies $\mathbf{c}[i] = \mathbf{x}[i]$ for all $i \in \mathsf{H} \setminus E$.*

In Appendix A.1 we prove that a t-robust conflict decodable code (as per Definition 1.9) satisfies the strong definition if and only if $d \geq 3t + 1$, as summarized in the following lemma.

Lemma 1.10. *Let $\mathcal{C}$ be an $(n, k, d)_q$ t-robust conflict decodable code. Then $\mathcal{C}$ satisfies the strong guarantees for bad inputs if and only if $d \geq 3t + 1$.*

1.3.1 Inefficient Robust-Decoding

It turns out that, just like in (non-robust) conflict decodable codes, the construction of robust conflict decodable codes can be reduced to the construction of a conflict checkable code that satisfies local-to-global consistency. However, unlike the case of non-robust decoding, here the generic conflict-decoder is not necessarily efficient: it is required to find all t-vertex covers in a graph, a task that, under standard computational complexity assumptions, takes time exponential in $t = \Omega(n)$ [CJ03, IPZ01].

Lemma 1.11. *Let $\mathcal{C}$ be an $(n, k, d)_q$ conflict checkable code that satisfies local-to-global consistency. Then $\mathcal{C}$ is t-robust conflict decodable code for $t = \lfloor (d-1)/2 \rfloor$.*

At a high level, we let the conflict-decoder find all vertex covers of size at most t, and output them as the explanations. This ensures that validity of explanations holds, and if the inputs are good then at least one explanation contains only corrupt servers, as they form a vertex cover of size at most t. One can verify that the rest of the requirements follow as well, by using the fact that $\mathcal{C}$ satisfies local-to-global consistency. The full proof is deferred to Appendix A.

In the appendix we also prove that the conditions in Lemma 1.11 are in fact necessary. Namely, for every $0 \le t \le n$, if a code $\mathcal{C}$ is t-robust conflict decodable then (1) it must be conflict checkable and t-conflict decodable; (2) the best achievable robustness t is $(d-1)/2$ where d is the code's distance; and (3) in such a case the code must also satisfy the local-to-global consistency property. By combining this with Theorem 1.8 we derive a Singleton-type bound for comparison-based t-robust conflict decodable code.

Lemma 1.12. *Let $\mathcal{C}$ be an $(n, k, d)_q$ comparison-based t-robust conflict decodable code, with $1 < d < n$ and $1 \le t \le \lfloor (d-1)/2 \rfloor$. Then $k \le (n - 2t + 1)/2$.*

Proof. Since $d \ge 2t + 1$, we can think of $\mathcal{C}$ as an $(n, k, d' := 2t + 1)_q$ comparison-based t-robust conflict decodable code. This code is conflict chekcable (Lemma A.2) and it satisfies local-to-global consistency with respect to d' (Lemma A.4), and so, by Theorem 1.8 it holds that $k \le (n-d'+2)/2 = (n - 2t + 1)/2$, as required. $\square$

Recall that the bivariate code $\mathcal{C}_{\mathsf{bivariate}}$ is a linear $[n, k = (\ell + 2)/2, d = n - \ell]_{q=p^{\ell+1}}$, conflict checkable code with local-to-global consistency, and is therefore comparison-based t-robust conflict decodable code with robustness $t = \lfloor (n - \ell - 1)/2 \rfloor$. This is optimal by Lemma 1.12. (When d is odd, $k = (n - 2t + 1)/2$, and when d is even $k = (n - 2t)/2$.)

Remark 1.13 (Efficient conflict-decoder for sub-optimal parameters and the relation to secret-sharing)**.** *While $\mathcal{C}_{\mathsf{bivariate}}$ provides an optimal construction, its generic conflict-decoder is inefficient and requires time* $\mathrm{poly}(2^t, n)$*, as discussed above. However, it turns out that if we think of $\mathcal{C}$ as a t-robust conflict decodable code for a* sub-optimal *threshold $t = \lfloor (n - \ell - 1)/3 \rfloor$, then we can obtain a code with a conflict-decoder that runs in time* polynomial *in n and t. (See Section 4.3.4 for more details.)*

Unfortunately, such sub-optimal dependency between the rate and the robustness falls short of providing the desired applications. More accurately, our cryptographic applications require efficient t-robust decoding with robustness as good as $t = \lfloor (n - \ell - 1)/2 \rfloor$ where ℓ is the individual degree of the underlying polynomials. Roughly speaking, we will use the code to construct a secret sharing scheme (similarly to Shamir's scheme [Sha79]) and so the degree parameter ℓ will correspond to the privacy threshold. (A secret will be embedded in a codeword such that any group of ℓ servers learn nothing on the secret.) Since we will be interested in the setting where the adversary corrupts up to $n/3$ servers, we will have to take $\ell, t \ge n/3$, which prevents us from using the sub-optimal version mentioned above.

1.3.2 Quasipolynomial-Time Conflict-Decoder from Trivariate Polynomials

Let $\mathbb{F}_p$ be a finite field of size $p \geq n$, let $1, \ldots, n$ be n distinct field elements, and let $1 \leq \ell < n$ be an integer. Consider the code,

$$\mathcal{C}_{\mathsf{trivariate}} = \left\{ (F(x,y,1), \ldots, F(x,y,n)) \;\middle|\; \begin{array}{c} F \text{ is a symmetric trivariate polynomial} \\ \text{of degree at most } \ell \text{ in each variable} \end{array} \right\},$$

where every codeword $\mathbf{c} \in \mathcal{C}_{\mathsf{trivariate}}$ corresponds to some symmetric trivariate polynomial $F(x,y,z)$ of degree at most ℓ in each variable, and the ith entry of $\mathbf{c}$ is the symmetric bivariate polynomial $\mathbf{c}[i] = F(x,y,i)$. In Section 4.3.5 we prove the following theorem.

Theorem 1.14. *The code $\mathcal{C}_{\mathsf{trivariate}}$ is an $[n, k = (\ell+3)/3, d = n - \ell]_q$ comparison-based t-robust conflict decodable code for $t = \lfloor (n - \ell - 1)/2 \rfloor$, with alphabet $q = p^{(\ell+2)(\ell+1)/2}$ and a conflict-decoder algorithm that runs in time $t^{O(\log t)} \cdot \mathrm{poly}(n)$.*

While the generic conflict-decoder has exponential dependency in t, the conflict-decoder of $\mathcal{C}_{\mathsf{trivariate}}$ has only quasipolynomial dependency in t, which is an almost-exponential improvement. Following Remark 1.13, we mention that the code $\mathcal{C}_{\mathsf{trivariate}}$ can be used for $(\ell+1)$-out-of-n secret sharing. To embed a secret s in a codeword, pick a random symmetric trivariate polynomial $F(x,y,z)$ of degree at most ℓ in each variable conditioned on $F(0,0,0) = s$, and distribute the corresponding codeword $\mathbf{c}$ by giving the symbol $\mathbf{c}[i]$ to the i-th server.

The conflict-decoder. We continue with a high-level description of the conflict-decoder. Recall that, given the conflict graph K, the generic inefficient conflict-decoder of Lemma 1.11 simply outputs all t-vertex covers of K as the explanations. To obtain an efficient conflict-decoder, we observe that the conflict graph is structured. Specifically, the key observation is that if a pair of honest servers, i and j, are conflicted, then the set of honest servers that are conflicted with either i or j is large and contains at least $(n-t) - \ell$ honest servers (including i and j). Indeed, every honest server k that is consistent with both i and j, induces some partial agreement between i and j of the form $f_i(k,j) = f_j(k,i)$ where f_i and f_j are the symmetric bivariate polynomials that are being held by the ith and jth servers, respectively. Thus, if there are $\ell+1$ honest servers that are consistent with both i and j, then i and j cannot be conflicted (since their residual degree-ℓ univariate polynomials, $f_i(\cdot, j)$ and $f_j(\cdot, i)$, agree). Getting back to the conflict graph, if an edge (i,j) appears between two honest servers, then their the joint neighborhood $|N(i) \cup N(j)|$ must be large, i.e.,

$$|N(i) \cup N(j)| \geq n - t - \ell = d - t \geq t + 1. \tag{1}$$

Based on this observation, we start by removing all edges (i,j) that do not satisfy (1) and derive a subgraph K'. Since we remove only edges that are incident to at least one corrupt server, we can focus on K'. (The requirements from the explanations address only the edges between honest parties.) After this "cleaning" process we are left with a graph that each of its edges has large neighborhood as in (1); we refer to such a graph as *t-edge-neighborhood graph*. In Section 3 we show that, by using a variant of the classic search-tree algorithm, it is possible to find *all* t-vertex covers in such graphs in time $t^{O(\log t)} \cdot \mathrm{poly}(n)$. Roughly speaking, we repeatedly choose a vertex of degree at least $(t+1)/2$ and recursively branch by placing either the vertex or all its neighbours into the vertex cover. The latter step leads to t'-edge-neighborhood graph in which we need to find all t'-vertex covers for $t' \leq (t+1)/2$. Therefore each path in the recursion tree can contain at most $O(\log t)$ such steps, which essentially implies the desired bound. As an additional result, we also show how to find a $(1+\epsilon)$-approximation of t-vertex covers in *polynomial time*, which will be important for our applications. The question of finding an exact vertex cover in polynomial-time algorithm in such graphs remains as an interesting open question.

Remark 1.15 (Comparison to Raz-Safra and Ben-Sasson-Sudan)**.** *Closely related arguments appear in the literature of probabilistic checkable proofs and locally testable codes codes [RS97] (see also the work of [BSS04] in the context of tensoring). Specifically, the a closely-related variant of t-edge neighborhood graph property was employed in these works to derive a better* soundness error *of a local test, i.e., to derive better information-theoretic bounds on the structure of the code. In contrast, we use t-edge neighborhood graph in order to reduce the* computational complexity *of the conflict-decoder, i.e., we use this extra combinatorial structure in order to obtain an* algorithmic result*.*

1.4 Application: The Round Complexity of Secure Multiparty Computation

In secure multiparty computation (MPC) with *information-theoretic security* there are n parties who wish to compute a function of their joint inputs at the presence of a *computationally-unbounded* active (aka Byzantine or malicious) rushing adversary that controls up to t of the parties. We follow the standard convention [BGW88] and assume that each pair of parties is connected by a secure and authenticated point-to-point channel, and that all parties have access to a common broadcast channel.

Applebaum et al. [AKP20] provided a three-round protocol for general MPC in the regime of *strong honest majority*, where $t \leq (n-1)/3$. This setting is of special interest since three rounds are necessary for any non-trivial resiliency threshold ($t \geq 2$) [GIKR02], and since the best achievable threshold in three rounds is $t \leq (n-1)/3$ [AKP20]. However, the protocol of [AKP20] has *exponential dependency* on the number of corrupt parties t. (These results are still meaningful since their protocol is secure even against a computationally-unbounded adversary.)

Our results. We note that the core combinatorial ingredient in the above construction (as well as in previous ones) is a robust conflict decodable code. Indeed, [AKP20] implicitly use the code $\mathcal{C}_{\mathsf{bivariate}}$ that supports only exponential-time robust conflict decoding, and accordingly suffer from exponential overhead. In this work, we show that by replacing $\mathcal{C}_{\mathsf{bivariate}}$ with $\mathcal{C}_{\mathsf{trivariate}}$, we can obtain a protocol that has quasipolynomial dependency on n. Formally, we prove the following theorem.

Theorem 1.16. *Every n-party functionality $\mathcal{F}$, represented as a boolean circuit of size s and depth d, can be realized by a 3-round protocol that provides statistical security against a static, active, rushing adversary corrupting up to $t < n/3$ of the parties. The complexity of the protocol is* $\mathrm{poly}(t^{\log t}, n, s, 2^d)$.

As in all known constructions of constant-round information-theoretic MPC, our protocol has an exponential dependency on the depth d of the circuit. Getting rid of this dependency, even in weaker adversarial models (e.g., passive adversary and resiliency of $t = 1$), is a famous open problem that goes back to [BMR90].

In addition, for almost-optimal resiliency of $t \leq n/3.01$, we can obtain protocol that has *polynomial* dependency on n. Formally, we prove the following theorem.

Theorem 1.17. *For every constant $\epsilon > 0$, every n-party functionality $\mathcal{F}$, represented as a boolean circuit of size s and depth d, can be realized by a 3-round protocol that provides statistical security against a static, active, rushing adversary corrupting up to t of the parties, where $t \leq n/(3+\epsilon)$. The complexity of the protocol is* $\mathrm{poly}(n, s, 2^d)$.

Our main technical contribution is a 2-round protocol for statistically-secure verifiable secret sharing (VSS), that has quasipolynomial dependency on t for optimal resiliency, and polynomial dependency on t for almost-optimal resiliency. We continue with a toy version, that highlights the main ideas in the construction.

Toy version: very-weak VSS. For simplicity, let us consider the following specialized version of VSS. A distinguished player, called the dealer, holds a symmetric trivariate polynomial $F(x, y, z)$ of degree at most t in each variable, and wants to let the ith party learn the bivariate polynomial $F(x, y, i)$, and nothing else. We require the following weak correctness property, that should hold even if the dealer and up to t parties are corrupted: At the end of the protocol there exists a subset I of $2t + 1$ parties so that every $i \in I$ outputs $F'(x, y, i)$ for some degree-t polynomial F' that is consistent with the dealer's polynomial if the dealer is honest. The rest of the honest parties (outside I) output a special failure symbol $\perp$. Our goal is to design a secure 2-round protocol for this task.

The polynomial $F(x, y, z)$ corresponds to a codeword $\mathbf{c}$ of the $[n, k = (t + 3)/3, d = n - t]_q$ t-robust conflict decodable code $\mathcal{C}_{\mathsf{trivariate}}$, where for the robustness parameter we used the fact that $n \geq 3t + 1$ so $(d - 1)/2 \geq t$. As we want to let the ith party learn $\mathbf{c}[i] = F(x, y, i)$, in the first round of the protocol we simply let the dealer send $\mathbf{c}[i]$ to i, and denote by $f_i(x, y)$ the bivariate polynomial that i received. From now on, we think of the ith party as the ith server that holds $\mathbf{c}[i]$, and of the corrupt parties as the corrupt servers.

In the second round our goal is to perform a secure public consistency check among the parties in order to obtain the conflict graph. To do so, we strongly use the fact that $\mathcal{C}_{\mathsf{trivariate}}$ is a *comparison-based* code, and let every pair of parties (i, j) do the following: (1) in the first round, we let i and j exchange a random univariate degree-t polynomial $r_{i,j}(x)$, that will be used as a one-time pad; and (2) in the second round, the i-th party broadcasts the univariate polynomials $b_{i,j}(x) := f_i(x, j) + r_{i,j}(x)$, and the jth party broadcasts $b_{j,i}(x) := f_j(x, i) + r_{i,j}(x)$. Given these values every party can tell whether these two parties are in conflict or not without learning anything on the actual content of their polynomials. Indeed, here we see the importance of comparison-codes: they allow to publish consistency check in a *secure and round-efficient* way! Of course, if one of the parties, i or j, is corrupt, she can effectively decide whether to generate a conflict or not.

After the second round, the parties locally compute the conflict graph, execute the conflict-decoder of $\mathcal{C}_{\mathsf{trivariate}}$, choose some canonical explanation E of size at most t, and let $I := [n] \setminus E$ be the complement set. If no explanation exists (which, by guarantees for good inputs, occurs only if the dealer is corrupt) then the parties conclude that the dealer is corrupt, and output some default polynomial (say, the all-zero polynomial). Otherwise every $i \in I$ outputs $f_i(x, y)$ while every $i \in E$ output $\perp$. Observe that $|I| = n - |E| \geq 2t + 1$, and that the guarantees of the robust code imply that even if the dealer is corrupt, the polynomials $f_i(x, y)$ of all honest parties in I uniquely define some symmetric trivariate polynomial $F(x, y, z)$ of degree at most t in each variable. We mention that in previous works [AKP20], that were implicitly based on the code $\mathcal{C}_{\mathsf{bivariate}}$, this step required exponential time because of the use of the generic conflict-decoder; in contrast, by using the efficient decoder of $\mathcal{C}_{\mathsf{trivariate}}$ we can obtain a quasipolynomial-time protocol!

In order to obtain a full-fledged VSS, we further exploit the guarantees of the robust code. For example, we need to use the fact that if a the dealer is honest, then there exists some explanation that contains only corrupt parties. See Section 5 for full details. Let us mention that the use of trivariate polynomials (or more generally codes with efficient robust conflict-decoder) is new in this context, and may open the door to additional applications. Indeed, a followup work of Abraham, Asharov, and Patra already employed our tools to derive new results in asynchronous secret sharing [AAP23].

Organization. The rest of the paper is organised as follows. Section 2 is devoted to the construction of an almost-MDS conflict checkable code. In Section 3 we study the notion of t-edge-neighborhood graphs and present efficient algorithms for finding all vertex covers in such graphs. Comparison-based codes are studied in Section 4 including Singleton-type bounds (Section 4.1), the relation to linear codes (Section 4.2), and a general framework for the construction of comparison-based robust conflict decodable codes from any linear MDS codes (Section 4.3). Finally, in Section 5 we present the applications of robust conflict decodable codes to secure multiparty computation.

Acknowledgement. We thank Arpita Patra for various discussions on verifiable secret sharing, Dana Ron for pointing us to the work of Alon [Alo02], and Oded Goldreich for comments about the relation to locally testable codes. We are especially grateful to Irit Dinur for sharing with us her insights about locally testable codes, and for pointing out the relation to [RS97] and [BSS04].

2 Construction of Almost-Optimal Conflict Checkable Codes

In this section we prove Theorem 1.3, that we repeat here.

Theorem 2.1 (Theorem 1.3 restated)**.** *For every integers $n \geq 3$ and $2 \leq d \leq n-1$, and every $\epsilon > 0$, there exists an integer $q = q(n, d, \epsilon)$ for which there exists an $(n, k, d)_q$ conflict checkable code for $k \geq n-d+1-\epsilon$.*

To prove the theorem, we first need the following lemma.

Lemma 2.2 (Main lemma)**.** *For every integers $n \geq 3$ and $1 \leq t \leq n-2$, and every $\epsilon > 0$, there exists an integer $M = M(n, t, \epsilon)$, such that for any integer $m \geq M$ there exists a set $\mathcal{F}$ of degree-t polynomials with coefficients in $\{0, \ldots, m\}$, of size at least*

$$|\mathcal{F}| \geq m^{t+1-\epsilon}$$

so that for every indices $\eta_1, \ldots, \eta_{t+2} \in [n]$, every elements $y_1, \ldots, y_{t+2} \in \mathbb{Z}$, and every $\binom{t+2}{2}$ polynomials $(f_{\ell,r})_{1 \leq \ell < r \leq t+2}$ in $\mathcal{F}$ that are not necessarily distinct, where $f_{\ell,r}$ satisfies $f_{\ell,r}(\eta_\ell) = y_\ell$ and $f_{\ell,r}(\eta_r) = y_r$, it holds that there exists a degree-t polynomial $G(x)$ such that $G(\eta_i) = y_i$ for all $i \in [1, \ldots, t+2]$.

Lemma 2.2 is proved in Section 2.1. We continue with the proof of Theorem 2.1 given the lemma.

Proof of Theorem 2.1. Let $t = n - d$ and $\epsilon' = \epsilon/2$, and let M be the integer promised by Lemma 2.2 when applied with (n, t, ϵ'), let $m \geq M$ be an integer that will be set later, and let $\mathcal{F}$ be the promised set of degree-t polynomials with coefficients in $\{0, \ldots, m\}$ of size at least $m^{t+1-\epsilon'}$. Let $q = (m + 1) \cdot (t+1) \cdot n^t$, and consider the n-partite graph $G = (V_1, \ldots, V_n, E)$, where $|V_i| = q$, and the edges are defined as follows: for every $f \in \mathcal{F}$ create a clique among the vertices $(i, f(i))_{i \in [n]}$. Let $\mathcal{C}$ be the code defined as follows: a word $\mathbf{c} \in [q]^n$ is in $\mathcal{C}$ if and only if the vertices $(i, \mathbf{c}[i])_{i \in [n]}$ form a clique in G. Then by Claim 1.2 the code $\mathcal{C}$ is a conflict checkable code.

We continue by proving that $\mathcal{C}$ is an $(n, k, d)_q$ conflict checkable code for $k \geq n - d + 1 - \epsilon$. Observe that $\mathcal{C}$ has length n and alphabet of size q, and therefore it only remains to analyze the dimension k and the distance d.

The dimension. Since all polynomials in $\mathcal{F}$ are of degree $t \leq n - 2$, every two polynomials may agree on at most $n - 2$ points, and therefore those polynomials create distinct codewords, so $|\mathcal{C}| \geq |\mathcal{F}|$. Observe that $m \geq \frac{q}{2n^t(t+1)}$, and the dimension of the code is

$$k \geq \log_q |\mathcal{F}| \geq \frac{\log m^{t+1-\epsilon'}}{\log q} \geq (t + 1 - \epsilon') - (t + 1 - \epsilon') \frac{\log(2n^t(t+1))}{\log q}.$$

Recall that $q = (m + 1) \cdot (t + 1) \cdot n^t$ and so, by taking m to be large enough, we can ensure that $(t + 1 - \epsilon') \frac{\log(2n^t(t+1))}{\log q} \leq \epsilon/2$, which implies that $k \geq t + 1 - \epsilon$, as required.

The distance. Let $\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C}$ be two distinct codewords. If these codewords correspond to a pair of polynomials $f_1, f_2 \in \mathcal{F}$ then they can agree on at most t locations (since the polynomials are of degree t) and so they must be of distance at least $d = n - t$, as required. We extend this argument for general codewords (that are induced by cliques in G) by showing that every codeword in $\mathcal{C}$ corresponds to an evaluation of some degree t polynomial G (that may not be a member of $\mathcal{F}$) and every pair of distinct codewords can agree on at most t locations. Details follow.

Fix a codeword $\mathbf{c} \in \mathcal{C}$. By definition, the vertices $(i, \mathbf{c}[i])_{i \in [n]}$ form a clique. For every $1 \leq \ell < r \leq n$, and every edge $((\ell, \mathbf{c}[\ell]), (r, \mathbf{c}[r]))$ there exists a polynomial $f_{\ell,r}(x) \in \mathcal{F}$ that satisfies $f_{\ell,r}(\ell) = \mathbf{c}[\ell]$ and $f_{\ell,r}(r) = \mathbf{c}[r]$. Consider the sub-clique $(i, \mathbf{c}[i])_{i \in [t+2]}$, and observe that, by Lemma 2.2, there exists a degree-t polynomial $G(x)$ that satisfies $G(i) = \mathbf{c}[i]$ for all $i \in [t + 2]$. In addition, for every $j > t + 2$, let $S_j := \{1, \ldots, t + 1\} \cup \{j\}$, and note that by Lemma 2.2, there exists a degree-t polynomial $G_j(x)$ that satisfies $G_j(i) = \mathbf{c}[i]$ for all $i \in S_j$. Therefore G_j and G agree on at least $t + 1$ points, so necessarily $G_j = G$. We conclude that $G(i) = \mathbf{c}[i]$ for all $i \in [n]$, as required. This concludes the proof of the theorem. $\square$

2.1 Proof of Lemma 2.2

Let $d, (B_i)_{i \in [t]}, (B_{i,i'})_{1 \leq i < i' \leq t}$, M and $m \geq M$ be integers that will be determined later, and let $L := \left\lfloor \frac{\log m}{\log d} \right\rfloor - 1$. Define

$$\mathcal{F} := \left\{ a_0 + a_1 x + \ldots + a_t x^t \;\middle|\; \begin{array}{l} a_0 \in \{0, \ldots, m\} \\ \forall i \in [t],\ a_i = \sum_{j=0}^{L} \alpha_{i,j} \cdot d^j, \text{ where } 0 \leq \alpha_{i,j} < \frac{d}{2n^t \cdot (n-1) \cdot t^2} \text{ is an integer,} \\ \forall i \in [t],\ \sum_{j=0}^{L} \alpha_{i,j}^2 = B_i, \\ \forall 1 \leq i < i' \leq t,\ \sum_{j=0}^{L} \alpha_{i,j} \cdot \alpha_{i',j} = B_{i,i'}. \end{array} \right\}.$$

For every $f \in \mathcal{F}$ such that $f(x) = \sum_{i=0}^{t} a_i x^i$, and every $i \in [t]$, it holds that

$$a_i = \sum_{j=0}^{L} \alpha_{i,j} \cdot d^j \leq \sum_{j=0}^{L} (d-1) \cdot d^t = (d-1) \cdot \frac{d^{L+1} - 1}{d - 1} \leq d^{\log_d(m)} = m,$$

so all the coefficients of f are indeed in $\{0,\ldots,m\}$. For $i\in[t]$, it will be convenient to think of a_i as the vector $(\alpha_{i,0},\ldots,\alpha_{i,L})$ that corresponds to its unique representation in base d. Our goal is to prove that the set $\mathcal{F}$ satisfies the lemma, and the rest of the proof is organized as follows. First, we deal with the size of the set $\mathcal{F}$ and prove the following claim.

Claim 2.3 (Size of $\mathcal{F}$)**.** *There exists a choice of $d,(B_i)_{i\in[t]},(B_{i,i'})_{1\le i<i'\le t}$ and M such that for every integer $m\ge M$ the set $\mathcal{F}$ has size at least $|\mathcal{F}|\ge m^{t+1-\epsilon}$.*

Claim 2.3 is proved in Section 2.1.1. Fix $d,(B_i)_{i\in[t]},(B_{i,i'})_{1\le i<i'\le t}$ and $m\ge M$ according to Claim 2.3. Fix any $\eta_1<\eta_2<\ldots<\eta_{t+2}$, any elements $y_1,\ldots,y_{t+2}$, and any $\binom{t+2}{2}$ polynomials $(f_{\ell,r})_{1\le\ell<r\le t+2}$ in $\mathcal{F}$, where $f_{\ell,r}$ satisfies $f_{\ell,r}(\eta_\ell)=y_\ell$ and $f_{\ell,r}(\eta_r)=y_r$. Denote $f_{\ell,r}(x)=\sum_{i=0}^{t}a_i^{\ell,r}\cdot x^i$, and for every $i\in[t]$ denote $a_i^{\ell,r}=\sum_{j=0}^{L}\alpha_{i,j}^{\ell,r}\cdot d^i$. Then, it only remains to prove the following claim.

Claim 2.4. *There exists a degree-t polynomial $G(x)$ such that $G(\eta_i)=y_i$ for all $i\in[1,\ldots,t+2]$.*

The rest of the section is organized as follows. In Section 2.1.1 we prove Claim 2.3. Then, in Section 2.1.2 we analyse some properties that the (vector-representation of the) coefficients $a_i^{\ell,r}$ satisfy. Then, in Section 2.1.3 we use those properties in order to prove Claim 2.4. At a high level, we do so by interpolating the $t+2$ points $(\eta_1,y_1),\ldots,(\eta_{t+2},y_{t+2})$ to obtain a polynomial $G(x)=g_0+g_1x+\ldots+g_{t+1}x^{t+1}$ that satisfies $G(\eta_i)=y_i$ for all $i\in[t+2]$, and then we prove that $g_{t+1}=0$ by using the vector representation of the coefficients.

2.1.1 Proof of Claim 2.3: The Size of $\mathcal{F}$

In this section we prove Claim 2.3. There are $m+1$ potential values for a_0, and at least $\left\lfloor\frac{d}{2n^{n+2}}\right\rfloor^{L+1}$ potential values for a_i for any $i>0$. Therefore, the number of potential polynomials is at least

$$m\cdot\left\lfloor\frac{d}{2n^{n+2}}\right\rfloor^{(L+1)\cdot t}\ge\frac{m^{t+1}}{d^{2n}\cdot(2n^{n+2})^{(L+1)n}}.$$

For each of the B's, the number of potential values is at most $(L+1)\left(\frac{d}{2n^t\cdot(n-1)\cdot t^2}\right)^2\le(L+1)d^2$, and the number of B's is $t+\binom{t}{2}\le t^2\le n^2$. Therefore, there exists a choice of the B's so that the set $\mathcal{F}$ has size at least

$$|\mathcal{F}|\ge\frac{m^{t+1}}{d^{2n}\cdot(2n^{n+2})^{(L+1)n}}\cdot\left(\frac{1}{(L+1)d^2}\right)^{n^2}.$$

Set $d=2^{\sqrt{\log m}}$ to obtain

$$|\mathcal{F}|\ge\frac{m^{t+1}}{2^{14n^2\log n\sqrt{\log m}}}=m^{t+1-\frac{14n^2\log n\sqrt{\log m}}{\log m}}=m^{t+1-\frac{14n^2\log n}{\sqrt{\log m}}},$$

and therefore, for $M:=\lceil 2^{(14n^2\log n)/\epsilon)^2}\rceil$ and every $m\ge M$ we have that $|\mathcal{F}|\ge m^{t+1-\epsilon}$, as required. This concludes the proof of Claim 2.3.

2.1.2 Some Basic Properties

For every $1 \le \ell < r \le t+2$ define $b^{\ell,r} := \sum_{i=1}^{t} a_i^{\ell,r} \cdot \sum_{k=0}^{i-1} \eta_\ell^k \cdot \eta_r^{i-1-k}$. As we will see, $b^{\ell,r}$ appears naturally in the interpolation of the $t+2$ points $(\eta_1, y_1), \dots, (\eta_{t+2}, y_{t+2})$, and therefore we dedicate this section to the study of (the vector representation of) those terms.

To simplify notation we let $\Gamma(\ell, r, i) := \sum_{k=0}^{i-1} \eta_\ell^k \cdot \eta_r^{i-1-k}$, so $b^{\ell,r} = \sum_{i=1}^{t} a_i^{\ell,r} \cdot \Gamma(\ell, r, i)$. The simple identity in Fact 2.5 implies that $((\eta_r)^i - (\eta_\ell)^i) = \Gamma(\ell, r, i) \cdot (\eta_r - \eta_\ell)$ for every $i \in [t]$.

Fact 2.5. *For every positive integer n it holds that $(x^n - y^n) = (x-y)(\sum_{i=0}^{n-1} x^i y^{n-1-i})$.*

We continue by analysing the vector representation of $b^{\ell,r}$.

The vector representation of $b^{\ell,r}$. For every $j \in \{0, \dots, L\}$ it holds that

$$\sum_{i=1}^{t} \alpha_{i,j}^{\ell,r} \cdot \sum_{k=0}^{i-1} \eta_\ell^k \cdot \eta_r^{i-1-k} < \frac{d}{2n^t \cdot (n-1) \cdot t^2} \cdot t^2 \cdot n^t = \frac{d}{2(n-1)} < d, \tag{2}$$

so the unique representation of $b^{\ell,r}$ in base d is given by the vector $\mathbf{v}^{\ell,r} := (\sum_{i=1}^{t} \alpha_{i,0}^{\ell,r} \cdot \Gamma(\ell, r, i), \dots, \sum_{i=1}^{t} \alpha_{i,L}^{\ell,r} \cdot \Gamma(\ell, r, i))$. The rest of this section is dedicated to the analysis of the norm and the inner product of the vectors $\mathbf{v}^{\ell,r}$: In Claim 2.6 we analyse *the ℓ^2-norm* of $\mathbf{v}^{\ell,r}$, and in Claim 2.7 we analyse *inner products* of the form $\mathbf{v}^{1,h} \cdot \mathbf{v}^{1,j}$ for $1 < h < j \le t+2$.

Claim 2.6 (ℓ^2-norm of $\mathbf{v}^{\ell,r}$)**.** *For every $1 \le \ell < r \le t+2$ it holds that*

$$\left\| \mathbf{v}^{\ell,r} \right\|_2^2 = \sum_{i=1}^{t} \Gamma(\ell, r, i)^2 \cdot B_i + 2 \sum_{1 \le k < i \le t} \Gamma(\ell, r, k) \Gamma(\ell, r, i) \cdot B_{k,i}.$$

Proof. A direct calculation shows that

$$\begin{aligned} \left\| \mathbf{v}^{\ell,r} \right\|_2^2 &= \sum_{j=0}^{L} \left(\sum_{i=1}^{t} \alpha_{i,j}^{\ell,r} \cdot \Gamma(\ell, r, i) \right)^2 \\ &= \sum_{i=1}^{t} \Gamma(\ell, r, i)^2 \cdot \sum_{j=0}^{L} (\alpha_{i,j}^{\ell,r})^2 + 2 \sum_{1 \le k < i \le t} \Gamma(\ell, r, k) \Gamma(\ell, r, i) \sum_{j=0}^{L} \alpha_{i,j}^{\ell,r} \cdot \alpha_{k,j}^{\ell,r} \\ &= \sum_{i=1}^{t} \Gamma(\ell, r, i)^2 \cdot B_i + 2 \sum_{1 \le k < i \le t} \Gamma(\ell, r, k) \Gamma(\ell, r, i) \cdot B_{k,i}, \end{aligned}$$

where we used the fact that $\sum_{j=0}^{L} (\alpha_{i,j}^{\ell,r})^2 = B_i$ for all $i \in [t]$, and $\sum_{j=0}^{L} \alpha_{i,j}^{\ell,r} \cdot \alpha_{k,j}^{\ell,r} = B_{k,i}$ for all $1 \le k < i \le t$. This completes the proof of the claim. $\square$

Claim 2.7 (Inner product)**.** *For every $1 < h < j \le t+2$ it holds that*

$$\mathbf{v}^{1,j} \cdot \mathbf{v}^{1,h} = \sum_{i=1}^{t} \Gamma(1, h, i) \Gamma(1, j, i) B_i + \sum_{1 \le k < i \le t} (\Gamma(1, h, k) \Gamma(1, j, i) + \Gamma(1, h, i) \Gamma(1, j, k)) B_{k,i}.$$

Proof. Fix any $1 < h < j \le t+2$. Observe that the following equations hold:

$$\begin{aligned} f_{1,h}(\eta_h) &= f_{h,j}(\eta_h) \\ f_{h,j}(\eta_j) &= f_{1,j}(\eta_j) \\ f_{1,j}(\eta_1) &= f_{1,h}(\eta_1). \end{aligned}$$

Summing up the three equations we obtain

$$\sum_{i=1}^{t} a_i^{1,j}((\eta_j)^i - (\eta_1)^i) = \sum_{i=1}^{t} a_i^{1,h}((\eta_h)^i - (\eta_1)^i) + \sum_{i=1}^{t} a_i^{h,j}((\eta_j)^i - (\eta_h)^i),$$

and using Fact 2.5 we obtain that

$$(\eta_j - \eta_1)b^{1,j} = (\eta_h - \eta_1)b^{1,h} + (\eta_j - \eta_h)b^{h,j}. \tag{3}$$

By Equation 2 it holds that $\mathbf{v}^{\ell,r}[i] < \frac{d}{2(n-1)}$ for every $1 \le \ell < r \le t$ and $i \in \{0, \dots, L\}$, so $(\eta_r - \eta_\ell)\mathbf{v}^{\ell,r}[i] < (n-1) \cdot \frac{d}{2(n-1)} = d/2$, and therefore $(\eta_h - \eta_1)\mathbf{v}^{1,h} + (\eta_j - \eta_h)\mathbf{v}^{h,j}$ is the unique representation in base d of the RHS of Equation 3, and $(\eta_j - \eta_1)\mathbf{v}^{1,j}$ is the unique representation in base d of the LHS of Equation 3. That is, we obtained a vector equality of the form

$$(\eta_j - \eta_1)\mathbf{v}^{1,j} = (\eta_h - \eta_1)\mathbf{v}^{1,h} + (\eta_j - \eta_h)\mathbf{v}^{h,j}. \tag{4}$$

Multiplying Equation 4 by $\mathbf{v}^{1,j}$, $\mathbf{v}^{1,h}$ and $\mathbf{v}^{h,j}$, we obtain the following three equalities

$$\begin{aligned} (\eta_j - \eta_1)\left\|\mathbf{v}^{1,j}\right\|_2^2 &= (\eta_h - \eta_1)\mathbf{v}^{1,j} \cdot \mathbf{v}^{1,h} + (\eta_j - \eta_h)\mathbf{v}^{1,j} \cdot \mathbf{v}^{h,j}, \\ (\eta_j - \eta_1)\mathbf{v}^{1,h} \cdot \mathbf{v}^{1,j} &= (\eta_h - \eta_1)\left\|\mathbf{v}^{1,h}\right\|_2^2 + (\eta_j - \eta_h)\mathbf{v}^{1,h} \cdot \mathbf{v}^{h,j}, \\ (\eta_j - \eta_1)\mathbf{v}^{h,j} \cdot \mathbf{v}^{1,j} &= (\eta_h - \eta_1)\mathbf{v}^{h,j} \cdot \mathbf{v}^{1,h} + (\eta_j - \eta_h)\left\|\mathbf{v}^{h,j}\right\|_2^2, \end{aligned}$$

that form three linear equations in the variables $(\mathbf{v}^{1,j} \cdot \mathbf{v}^{1,h})$, $(\mathbf{v}^{1,j} \cdot \mathbf{v}^{h,j})$, and $(\mathbf{v}^{1,h} \cdot \mathbf{v}^{h,j})$. The unique solution is given by

$$\begin{aligned} \mathbf{v}^{1,j} \cdot \mathbf{v}^{1,h} &= \frac{\eta_j - \eta_1}{2(\eta_h - \eta_1)}\left\|\mathbf{v}^{1,j}\right\|_2^2 + \frac{\eta_h - \eta_1}{2(\eta_j - \eta_1)}\left\|\mathbf{v}^{1,h}\right\|_2^2 - \frac{(\eta_j - \eta_h)^2}{2(\eta_h - \eta_1)(\eta_j - \eta_1)}\left\|\mathbf{v}^{h,j}\right\|_2^2 \\ \mathbf{v}^{1,j} \cdot \mathbf{v}^{h,j} &= \frac{\eta_j - \eta_1}{2(\eta_j - \eta_h)}\left\|\mathbf{v}^{1,j}\right\|_2^2 - \frac{(\eta_h - \eta_1)^2}{2(\eta_j - \eta_1)(\eta_j - \eta_h)}\left\|\mathbf{v}^{1,h}\right\|_2^2 + \frac{\eta_j - \eta_h}{2(\eta_j - \eta_1)}\left\|\mathbf{v}^{h,j}\right\|_2^2 \\ \mathbf{v}^{1,h} \cdot \mathbf{v}^{h,j} &= \frac{(\eta_j - \eta_1)^2}{2(\eta_j - \eta_h)(\eta_h - \eta_1)}\left\|\mathbf{v}^{1,j}\right\|_2^2 - \frac{\eta_h - \eta_1}{2(\eta_j - \eta_h)}\left\|\mathbf{v}^{1,h}\right\|_2^2 - \frac{\eta_j - \eta_h}{2(\eta_h - \eta_1)}\left\|\mathbf{v}^{h,j}\right\|_2^2. \end{aligned} \tag{5}$$

We continue by computing the term $\mathbf{v}^{1,j} \cdot \mathbf{v}^{1,h}$ by substituting the terms $\left\|\mathbf{v}^{1,j}\right\|_2^2$, $\left\|\mathbf{v}^{1,h}\right\|_2^2$ and $\left\|\mathbf{v}^{h,j}\right\|_2^2$ in Equation 5 using Claim 2.6. A direct calculation shows that for every $i \in [t]$, the coefficient of B_i is given by

$$\begin{aligned} &\frac{\eta_j - \eta_1}{2(\eta_h - \eta_1)}\Gamma(1,j,i)^2 + \frac{\eta_h - \eta_1}{2(\eta_j - \eta_1)}\Gamma(1,h,i)^2 - \frac{(\eta_j - \eta_h)^2}{2(\eta_h - \eta_1)(\eta_j - \eta_1)}\Gamma(h,j,i)^2 \\ &\quad = \frac{1}{2(\eta_h - \eta_1)(\eta_j - \eta_1)}\left(\left((\eta_j)^i - (\eta_1)^i\right)^2 + \left((\eta_h)^i - (\eta_1)^i\right)^2 - \left((\eta_j)^i - (\eta_h)^i\right)^2\right) \\ &\quad = \frac{((\eta_j)^i - (\eta_1)^i)((\eta_h)^i - (\eta_1)^i)}{(\eta_h - \eta_1)(\eta_j - \eta_1)} = \Gamma(1,h,i)\Gamma(1,j,i), \end{aligned}$$

where we used Fact 2.5. In addition, for $1 \le k < i \le t$, the coefficient of $B_{k,i}$ is given by

$$
\begin{aligned}
&\frac{\eta_j - \eta_1}{2(\eta_h - \eta_1)} \cdot 2\Gamma(1,j,k)\Gamma(1,j,i) + \frac{\eta_h - \eta_1}{2(\eta_j - \eta_1)} \cdot 2\Gamma(1,h,k)\Gamma(1,h,i) - \frac{(\eta_j - \eta_h)^2}{2(\eta_h - \eta_1)(\eta_j - \eta_1)} \cdot 2\Gamma(h,j,k)\Gamma(h,j,i) \\
&= \frac{((\eta_j)^k - (\eta_1)^k)((\eta_j)^i - (\eta_1)^i) + ((\eta_h)^k - (\eta_1)^k)((\eta_h)^i - (\eta_1)^i) - ((\eta_j)^k - (\eta_h)^k)((\eta_j)^i - (\eta_h)^i)}{(\eta_j - \eta_1)(\eta_h - \eta_1)} \\
&= \frac{((\eta_j)^i - (\eta_1)^i)((\eta_h)^k - (\eta_1)^k) + ((\eta_j)^k - (\eta_1)^k)((\eta_h)^i - (\eta_1)^i)}{(\eta_j - \eta_1)(\eta_h - \eta_1)} \\
&= \Gamma(1,h,k)\Gamma(1,j,i) + \Gamma(1,h,i)\Gamma(1,j,k),
\end{aligned}
$$

where we used Fact 2.5. Therefore,

$$
\mathbf{v}^{1,j} \cdot \mathbf{v}^{1,h} = \sum_{i=1}^{t} \Gamma(1,h,i)\Gamma(1,j,i)B_i + \sum_{1 \le k < i \le t} (\Gamma(1,h,k)\Gamma(1,j,i) + \Gamma(1,h,i)\Gamma(1,j,k))B_{k,i}.
$$

which completes the proof of the claim. □

2.1.3 Proof of Claim 2.4: Interpolation

Let $G(x) = g_0 + g_1 x + \ldots + g_{t+1}x^{t+1}$ be the polynomial obtained by interpolating the $t+2$ points $(\eta_1, y_1), \ldots, (\eta_{t+2}, y_{t+2})$. Formally, $G(x)$ is given by

$$
G(x) = \sum_{j=1}^{t+2} \left(y_j \cdot \prod_{\substack{1 \le h \le t+2 \\ h \ne j}} \frac{(x - \eta_h)}{(\eta_j - \eta_h)} \right), \quad \text{hence} \quad g_{t+1} = \sum_{j=1}^{t+2} \left(y_j \cdot \prod_{\substack{1 \le h \le t+2 \\ h \ne j}} \frac{1}{(\eta_j - \eta_h)} \right).
$$

The rest of the section is dedicated to proving that $g_{t+1} = 0$, which means that $G(x)$ has degree t. This involves a rather tedious (but straightforward) computation, using the machinery developed in the previous section.

Calculating the y_j's. We continue by calculating the y_j's. To simplify notation, for every $j \in [t+2]$ we denote $\pi(j) := \prod_{\substack{1 \le h \le t+2 \\ h \ne j}} \frac{1}{(\eta_j - \eta_h)}$. For $j = 1$ we have $y_1 = f_{1,2}(\eta_1) = \sum_{i=0}^{t} a_i^{1,2}(\eta_1)^i$, and for $j = 2$ we have $y_2 = f_{1,2}(\eta_2) = \sum_{i=0}^{t} a_i^{1,2}(\eta_2)^i$. For every $j > 2$ it holds that $y_j = f_{1,j}(\eta_j) = \sum_{i=0}^{t} a_i^{1,j}(\eta_j)^i$, and in addition it holds that $f_{1,j}(\eta_1) = f_{1,2}(\eta_1)$, so $\sum_{i=0}^{t} a_i^{1,2}(\eta_1)^i = \sum_{i=0}^{t} a_i^{1,j}(\eta_1)^i$, which means that $a_0^{1,j} = \sum_{i=0}^{t} a_i^{1,2}(\eta_1)^i - \sum_{i=1}^{t} a_i^{1,j}(\eta_1)^i$ and therefore

$$
y_j = \left(\sum_{i=0}^{t} a_i^{1,2}(\eta_1)^i - \sum_{i=1}^{t} a_i^{1,j}(\eta_1)^i \right) + \sum_{i=1}^{t} a_i^{1,j}(\eta_j)^i = \sum_{i=0}^{t} a_i^{1,2}(\eta_1)^i + \sum_{i=1}^{t} a_i^{1,j}\left((\eta_j)^i - (\eta_1)^i\right).
$$

We conclude that

$$
\begin{aligned}
g_{t+1} = &\left(\sum_{i=0}^{t} a_i^{1,2}(\eta_1)^i \right) \cdot \pi(1) + \left(\sum_{i=0}^{t} a_i^{1,2}(\eta_2)^i \right) \cdot \pi(2) \\
&+ \sum_{j=3}^{t+2} \left(\sum_{i=0}^{t} a_i^{1,2}(\eta_1)^i + \sum_{i=1}^{t} a_i^{1,j}\left((\eta_j)^i - (\eta_1)^i\right) \right) \cdot \pi(j).
\end{aligned} \tag{6}
$$

Calculating the coefficient of $a_j^{1,i}$. We continue by analysing the coefficient of $a_j^{1,i}$ in Equation 6. First, we prove the following claim.

Claim 2.8. *For every $i \in \{0, \dots, t\}$, it holds that $\sum_{j=1}^{t+2} (\eta_j)^i \cdot \pi(j) = 0$.*

Proof. Consider the interpolation over the $t+2$ points $(\eta_1, (\eta_1)^i), \dots, (\eta_{t+2}, (\eta_{t+2})^i)$, where we obtain a polynomial $H(x) = h_0 + h_1 x + \ldots + h_{t+1} x^{t+1}$, and note that $H(x) = x^i$, so $h_{t+1} = 0$. From interpolation we also obtain that

$$0 = h_{t+1} = \sum_{j=1}^{t+2} (\eta_j)^i \cdot \prod_{\substack{1 \le h \le t+2 \\ h \neq j}} \frac{1}{(\eta_j - \eta_h)} = \sum_{j=1}^{t+2} (\eta_j)^i \cdot \pi(j),$$

as required. This conclude the proof of the claim. □

It is not hard to see that the coefficient of $a_0^{1,2}$ is $\sum_{j=1}^{t+2} \pi(j)$, so from Claim 2.8 this coefficient is 0. We continue with the analysis of the coefficient of $a_i^{1,2}$ for $i > 0$. A direct calculation shows that the coefficient is $(\eta_1)^i \cdot \sum_{\substack{1 \le j \le t+2 \\ j \neq 2}} \pi(j) + (\eta_2)^i \cdot \pi(2)$. From Claim 2.8 we conclude that $\sum_{\substack{1 \le j \le t+2 \\ j \neq 2}} \pi(j) = -\pi(2)$, and therefore the coefficient is $\pi(2)((\eta_2)^i - (\eta_1)^i)$. Finally, for every $j \ge 3$ and $i \ge 1$ the coefficient of $a_i^{1,j}$ is $\pi(j)((\eta_j)^i - (\eta_1)^i)$. From Fact 2.5 we obtain that

$$g_{t+1} = \sum_{j=2}^{t+2} \left(\pi(j) \cdot (\eta_j - \eta_1) \cdot \sum_{i=1}^{t} a_i^{1,j} \cdot \sum_{k=0}^{i-1} \eta_j^k \cdot \eta_1^{i-1-k} \right) = \sum_{j=2}^{t+2} \pi(j) \cdot (\eta_j - \eta_1) \cdot b^{1,j}.$$

In order to prove that $g_{t+1} = 0$, we note that

$$g_{t+1} = \sum_{j=2}^{t+2} \left(\pi(j) \cdot (\eta_j - \eta_1) \cdot (\sum_{\ell=0}^{L} \mathbf{v}^{1,j}[\ell] \cdot d^\ell) \right) = \sum_{\ell=0}^{L} d^\ell \cdot \left(\sum_{j=2}^{t+2} \pi(j) \cdot (\eta_j - \eta_1) \mathbf{v}^{1,j}[\ell] \right).$$

Therefore, it is enough to prove that for every $\ell \in \{0, \dots, L\}$ it holds that $\sum_{j=2}^{t+2} \pi(j) \cdot (\eta_j - \eta_1) \mathbf{v}^{1,j}[\ell] = 0$. In order to do so, it is enough to prove that the ℓ^2-norm of the vector $\mathbf{v} := \sum_{j=2}^{t+2} \pi(j) \cdot (\eta_j - \eta_1) \mathbf{v}^{1,j}$ is 0.

The norm of v. Observe that

$$\|\mathbf{v}\|_2^2 = \sum_{h=2}^{t+2} \pi(h)^2 \cdot (\eta_h - \eta_1)^2 \left\| \mathbf{v}^{1,h} \right\|_2^2 + 2 \sum_{2 \le h < j \le t+2} \pi(h)\pi(j)(\eta_h - \eta_1)(\eta_j - \eta_1) \mathbf{v}^{1,h} \cdot \mathbf{v}^{1,j}. \tag{7}$$

Substitute the terms $\left\| \mathbf{v}^{1,h} \right\|_2^2$ and $\mathbf{v}^{1,h} \cdot \mathbf{v}1, j$ using Claim 2.6 and Claim 2.7. We continue by showing that the coefficient of every B_i and every $B_{k,i}$ in Equation 7 is 0, and therefore the norm is 0.

For every $1 \le i \le t$, the coefficient of B_i is given by

$$\sum_{h=2}^{t+2} \pi(h)^2 \cdot (\eta_h - \eta_1)^2 \Gamma(1,h,i)^2 + 2 \sum_{2 \le h < j \le t+2} \pi(h)\pi(j)(\eta_h - \eta_1)(\eta_j - \eta_1)\Gamma(1,h,i)\Gamma(1,j,i)$$

$$
\begin{aligned}
&= \sum_{h=2}^{t+2} \pi(h)^2 \cdot ((\eta_h)^i - (\eta_1)^i)^2 + 2 \sum_{2 \le h < j \le t+2} \pi(h)\pi(j)((\eta_h)^i - (\eta_1)^i)((\eta_j)^i - (\eta_1)^i) \\
&= \sum_{h=2}^{t+2} \pi(h)^2 \cdot ((\eta_h)^i - (\eta_1)^i)^2 + \sum_{2 \le h \le t+2} \sum_{\substack{2 \le j \le t+2 \\ j \ne h}} \pi(h)\pi(j)((\eta_h)^i - (\eta_1)^i)((\eta_j)^i - (\eta_1)^i) \\
&= \sum_{h=2}^{t+2} \left(\pi(h) \cdot ((\eta_h)^i - (\eta_1)^i) \left(\pi(h) \cdot ((\eta_h)^i - (\eta_1)^i) + \sum_{\substack{2 \le j \le t+2 \\ j \ne h}} \pi(j)((\eta_j)^i - (\eta_1)^i) \right) \right) \\
&= \sum_{h=2}^{t+2} \left(\pi(h) \cdot ((\eta_h)^i - (\eta_1)^i) \left(\sum_{2 \le j \le t+2} \pi(j)((\eta_j)^i - (\eta_1)^i) \right) \right) \\
&= \left(\sum_{2 \le j \le t+2} \pi(j)((\eta_j)^i - (\eta_1)^i) \right)^2 = \left(\left(\sum_{2 \le j \le t+2} \pi(j)(\eta_j)^i \right) - \left((\eta_1)^i \sum_{2 \le j \le t+2} \pi(j) \right) \right)^2 \\
&= \left(\sum_{1 \le j \le t+2} \pi(j)(\eta_j)^i \right)^2 = 0,
\end{aligned}
$$

where in the first equality we used Fact 2.5, and in the last two equalities we used Claim 2.8. In addition, for every $1 \le k < i \le t$, the coefficient of $B_{k,i}$ is given by

$$
\begin{aligned}
&\sum_{h=2}^{t+2} \pi(h)^2 \cdot (\eta_h - \eta_1)^2 \cdot 2\Gamma(1,h,k) \cdot \Gamma(1,h,i) \\
&\quad + 2 \sum_{2 \le h < j \le t+2} \pi(h)\pi(j)(\eta_h - \eta_1)(\eta_j - \eta_1)(\Gamma(1,h,k)\Gamma(1,j,i) + \Gamma(1,h,i)\Gamma(1,j,k)) \\
&= \sum_{h=2}^{t+2} \left(\pi(h)^2 \cdot ((\eta_h)^k - (\eta_1)^k)((\eta_h)^i - (\eta_1)^i) + \sum_{\substack{2 \le j \le t+2 \\ j \ne h}} \pi(h)\pi(j)((\eta_h)^k - (\eta_1)^k)((\eta_j)^i - (\eta_1)^i) \right) \\
&\quad + \sum_{h=2}^{t+2} \left(\pi(h)^2 \cdot ((\eta_h)^k - (\eta_1)^k)((\eta_h)^i - (\eta_1)^i) + \sum_{\substack{2 \le j \le t+2 \\ j \ne h}} \pi(h)\pi(j)((\eta_h)^i - (\eta_1)^i)((\eta_j)^k - (\eta_1)^k) \right) \\
&= \sum_{h=2}^{t+2} \left(\pi(h) \cdot ((\eta_h)^k - (\eta_1)^k) \left(\pi(h)((\eta_h)^i - (\eta_1)^i) + \sum_{\substack{2 \le j \le t+2 \\ j \ne h}} \pi(j)((\eta_j)^i - (\eta_1)^i) \right) \right) \\
&\quad + \sum_{h=2}^{t+2} \left(\pi(h) \cdot ((\eta_h)^i - (\eta_1)^i) \left(\pi(h)((\eta_h)^k - (\eta_1)^k) + \sum_{\substack{2 \le j \le t+2 \\ j \ne h}} \pi(j)((\eta_j)^k - (\eta_1)^k) \right) \right)
\end{aligned}
$$

$$
\begin{aligned}
&= 2\left(\sum_{h=2}^{t+2}\pi(h)\cdot((\eta_h)^i-(\eta_1)^i)\right)\left(\sum_{h=2}^{t+2}\pi(h)\cdot((\eta_h)^k-(\eta_1)^k)\right)\\
&= 2\left(\sum_{h=2}^{t+2}\pi(h)\cdot(\eta_h)^i-(\eta_1)^i\sum_{h=2}^{t+2}\pi(h)\right)\left(\sum_{h=2}^{t+2}\pi(h)\cdot(\eta_h)^k-(\eta_1)^k\sum_{h=2}^{t+2}\pi(h)\right)\\
&= 2\left(\sum_{1\le j\le t+2}\pi(j)(\eta_j)^i\right)\left(\sum_{1\le j\le t+2}\pi(j)(\eta_j)^k\right)=0,
\end{aligned}
$$

where in the first equality we used Fact 2.5, and in the last two equalities we used Claim 2.8. Therefore $g_{t+1}=0$, and $G(x)$ is a degree-t polynomial that satisfies $G(\eta_i)=y_i$ for every $i\in[t+2]$, as required. This concludes the proof of Claim 2.4, and the proof of Lemma 2.2.

3 t-Edge-Neighborhood Graphs

In this section we formally present the notion of t-edge-neighborhood graphs, together with efficient algorithms for finding vertex cover in such graphs. Let us begin with a formal definition.

Definition 3.1 (t-edge neighborhood graphs)**.** *Let $G=(V,E)$ be a graph with n vertices. For an integer $1\le t\le n-1$ we say that G is a t-*edge-neighborhood graph *if for every edge $(u,v)\in E$ it holds that*

$$|N(u)\cup N(v)|\ge t+1.$$

The rest of this section is organised as follows. In Section 3.1 we provide a quasipolynomial-time algorithm for finding all t-vertex covers in t-edge neighborhood graphs. In Section 3.2 we present a polynomial-time algorithm for finding all $(1+\epsilon)$-approximations of t-vertex covers in t-edge neighborhood graphs.

3.1 Quasipolynomial-Time Algorithm for Vertex Cover

In this section we show how to find all size-t vertex covers in t-edge-neighborhood graphs in time $t^{O(\log t)}\cdot\mathrm{poly}(n)$. Formally, our goal is to solve the following algorithmic problem.

- *Input:* A graph $G=(V,E)$ with n vertices and m edges, and an integer $t\in[n-1]$.
- *Promise:* The graph G is a t-edge-neighborhood graph.
- *Goal:* Find all t-vertex covers of G.

By finding all t-vertex covers of G we mean finding vertex covers $S_1,\dots,S_\ell$ of G, each of size at most t, so that for every vertex cover I of G of size at most t, there exists some S_i such that $S_i\subseteq I$.

High-level idea. Our algorithm is based on the search-tree approach. That is, at each step we take from the graph a vertex v that has maximal degree, and we note that for every vertex cover of G, either v is in the vertex cover, or $N(v)$ is in the vertex cover. Therefore, we try both cases: a right step on the tree means that we add v to the vertex cover, and remove v from the graph; a left step on the tree means that we add $N(v)$ to the vertex cover, and remove $N(v)$ from the graph. It is not hard to verify that this approach guarantees that we find all t-vertex covers.

For the running time, we use the fact that the graph is a t-edge-neighborhood graph. This means that in the first step, there must exist a vertex v whose degree is at least $(t+1)/2$, and, in addition, it is not hard to see that if we remove k vertices from the graph then the graph is a $(t-k)$-edge-neighborhood graph. Therefore, every path from root to leaf has length at most t, and the number of left steps in the path is at most $\lceil \log(t) \rceil + 1$, so the number of such paths is at most $\binom{t}{\lceil \log(t) \rceil + 1} = t^{O(\log(t))}$.

The search tree. We continue with a formal description of the search tree.

FullSearchTree

Consider the binary tree that is defined recursively as follows. The root is labelled by the tuple $(G, n, t, v, \varnothing)$, where v is some vertex of G that has maximal degree, and $\varnothing$ is the empty set. For every node u in the binary search tree with label (G', n', t', v', S'), the children of this node are defined as follows.

- If G' contains no edges then u is a leaf, and its label is changed to S'.
- Otherwise, if $t' = 0$ (and G' contains edges), then u is a leaf, and its label is changed to `Failure`.
- Otherwise $t' > 0$ and we split into cases.
  - If $d_{G'}(v') > t'$ (i.e., the degree of v' in G' is greater than t'), let H be the graph obtained from G' by removing the vertex v'. Then u has only a right son labelled with $(H, n'-1, t'-1, w, S' \cup \{v'\})$ where w is a vertex of maximum degree in H.
  - Otherwise $d_{G'}(v') \leq t'$. Let H_R be the graph obtained from G' by removing the vertex v', and let H_L be the graph obtained from G' by removing the vertices $N_{G'}(v')$ (i.e., the neighbors of v' in G'). Let w_R be a vertex of maximum degree of H_R and let w_L be a vertex of maximum degree of H_L. Then u has a right child labelled with $(H_R, n'-1, t'-1, w_R, S' \cup \{v'\})$, and a left child labelled with $(H_L, n' - d_{G'}(v'), t' - d_{G'}(v'), w_L, S' \cup N_{G'}(v'))$.

Figure 1: FullSearchTree

The algorithm. We continue with a description for an algorithm that finds all t-vertex covers in t-edge-neighborhood graphs, in time $t^{O(\log t)} \cdot \mathrm{poly}(n)$.

Algorithm ExactVC

Input: A graph $G = (V, E)$ with n vertices and m edges, and an integer $t \in [n-1]$.

Promise: The graph G is a t-edge-neighborhood graph.

The algorithm: The algorithm constructs the full search tree, defined in Figure 1, Let $S_1, \ldots, S_\ell$ be the labels of all the leaves that are not labelled with `Failure`. The algorithm outputs $S_1, \ldots, S_\ell$.

Figure 2: Algorithm ExactVC

Theorem 3.2. *Algorithm ExactVC, described in Figure 2, on input $G = (V, E)$ with n vertices and m edges, and an integer $t \in [n-1]$, where G is a t-edge-neighborhood graph, outputs all t-vertex covers of G in time $t^{O(\log t)} \cdot \mathrm{poly}(n)$.*

Proof. We first prove the correctness of the algorithm, and then analyse its running time.

Correctness. We need to prove that (1) for every leaf with label S, the set S is a vertex cover of size at most t, and (2) for every t-vertex cover I of G there exists a leaf with label S such that $S \subseteq I$. First we note that for every node in the search tree with label (G', n', t', v', S'), the graph G' was obtained from G by removing the vertices in the set S', the graph G' has $n' = n - |S|$ vertices, it holds that $|S'| \leq t$ and $t' = t - |S|$, and the vertex v' has maximal degree in G'. We therefore conclude that S' covers all the edges that were removed from G, and in particular, for every leaf with label S' it holds that S' has size at most t and it is a vertex cover, so (1) holds. For (2), let I be any vertex cover of size at most t, and consider the walk from root to leaf on the search tree, where at a node with label (G', n', t', v', S'), we turn right if $v' \in I$, and turn left otherwise. Then it is not hard to verify that at each step it holds that $S' \subseteq I$, and in particular, $S \subseteq I$, where S is the label of the corresponding leaf.

Running time. Let $T(n, t)$ be the size of the largest search tree among all search trees of graphs with n vertices that are t-edge-neighborhood graphs. Observe that for every graph with n vertices that is t-edge-neighborhood graph, the search tree can be constructed in time $\mathrm{poly}(T(n,t), n)$. Therefore, it is enough to prove that $T(n, t) = t^{O(\log t)}$.

We first observe that for every node in the search tree with label (G', n', t', v', S'), it holds that G' is a t'-edge-neighborhood graph, so v' has degree at least $\lceil (t'+1)/2 \rceil$ in G'. Therefore, from any path from root to leaf, the number of left steps can be at most $\lceil \log(t) \rceil + 1$. In addition, the total number of steps in such a path is at most t, and therefore the total number of such paths is $\binom{t}{\lceil \log(t) \rceil + 1} = t^{O(\log t)}$, and $T(n,t) \leq (t+1) \cdot t^{O(\log t)} = t^{O(\log t)}$. This completes the proof of the theorem.

□

3.2 Polynomial-Time $(1+\epsilon)$-Approximation for Vertex Cover

In this section we show how to find all size-$(1+\epsilon)t$ vertex covers, for $\epsilon > 0$, in t-edge-neighborhood graphs in time $t^{O(\log(1/\epsilon))} \cdot \mathrm{poly}(n)$. Formally, our goal is to solve the following algorithmic problem.

- *Input:* A graph $G = (V, E)$ with n vertices and m edges, an integer $t \in [n-1]$, and some $\epsilon > 0$.
- *Promise:* The graph G is a t-edge-neighborhood graph.
- *Goal:* Find all $(1+\epsilon)$-approximation of t-vertex covers of G.

By finding all $(1+\epsilon)$-approximation of t-vertex covers of G, we mean finding vertex covers $S_1, \ldots, S_\ell$ of G, each of size at most $(1+\epsilon)t$, so that for every vertex cover I of G of size at most t, there exists some S_i for which $|S_i \setminus I| \leq 2\epsilon t$.

High-level idea. We construct the same search-tree as in Section 3.1, but now we truncate the tree at every node u with label (G', n', t', v', S') so that $t' \leq \epsilon \cdot t$. This will guarantee that the size of the search tree will be at most $t^{O(\log(1/\epsilon))}$. For every leaf in the truncated search tree with label (G', n', t', v', S') we execute the 2-approximation algorithm for vertex cover on G' in order to find a vertex cover S'' of G'. If S'' has size at most $2t'$ then $S' \cup S''$ forms a vertex cover of size at most $(t - t') + 2t' = t + t' \leq (1 + \epsilon)t$, so we set the label of the leaf to $S' \cup S''$. Otherwise, if S'' has size more than $2t'$ then G' doesn't have a vertex cover of size t', and therefore we change the label of the leaf to `Failure`.

The search tree. We continue with a formal description of the search tree.

TruncatedSearchTree

Consider the binary tree that is defined recursively as follows. The root is labelled by the tuple $(G, n, t, v, \varnothing)$, where v is some vertex of G that has maximal degree, and $\varnothing$ is the empty set. For every node u in the binary search tree with label (G', n', t', v', S'), the children of this node are defined as follows.

- If G' contains no edges then u is a leaf, and its label is changed to S'.
- Otherwise, if $t' \leq \epsilon \cdot t$, then u is a leaf. Execute the 2-approximation algorithm on G' to obtain a vertex cover S''. If $|S''| \leq 2t'$ then change the label of u to be the set $S' \cup S''$. Otherwise, if $|S''| > 2t'$, change the label of u to be `Failure`.
- Otherwise $t' > \epsilon \cdot t$ and we split into cases.
  - If $d_{G'}(v') > t'$, let H be the graph obtained from G' by removing the vertex v'. Then u has only a right son labelled with $(H, n' - 1, t' - 1, w, S' \cup \{v'\})$ where w is a vertex of maximum degree in H.
  - Otherwise $d_{G'}(v') \leq t'$. Let H_R be the graph obtained from G' by removing the vertex v', and let H_L be the graph obtained from G' by removing the vertices $N_{G'}(v')$. Let w_R be a vertex of maximum degree of H_R and let w_L be a vertex of maximum degree of H_L. Then u has a right child labelled with $(H_R, n' - 1, t' - 1, w_R, S' \cup \{v'\})$, and a left child labelled with $(H_L, n' - d_{G'}(v'), t' - d_{G'}(v'), w_L, S' \cup N_{G'}(v'))$.

Figure 3: TruncatedSearchTree

The algorithm. We continue with a description for an algorithm that finds all $(1 + \epsilon)$-approximation of t-vertex covers in t-edge-neighborhood graphs, in time $t^{O(\log(1/\epsilon))} \cdot \mathrm{poly}(n)$.

Algorithm ApproxVC

Input: A graph $G = (V, E)$ with n vertices and m edges, an integer $t \in [n - 1]$, a value $\epsilon > 0$.

Promise: The graph G is a t-edge-neighborhood graph.

The algorithm: The algorithm constructs the truncated search tree, defined in Figure 3. Let $S_1, \ldots, S_\ell$ be the labels of all the leaves that are not labelled with `Failure`. The algorithm outputs $S_1, \ldots, S_\ell$.

Figure 4: Algorithm ApproxVC

Theorem 3.3. *Algorithm ApproxVC, described in Figure 4, on input $G = (V, E)$ with n vertices and m edges, an integer $t \in [n-1]$, and a value $\epsilon > 0$, where G is a t-edge-neighborhood graph, outputs all $(1+\epsilon)$-approximations of t-vertex covers of G in time $t^{O(\log(1/\epsilon))} \cdot \mathrm{poly}(n)$.*

Proof. We first prove the correctness of the algorithm, and then analyse its running time.

Correctness. We need to prove that (1) for every leaf with label S, the set S is a vertex cover of size at most $(1+\epsilon)t$, and (2) for every t-vertex cover I of G, there exists a leaf with label S such that $|S \setminus I| \le 2\epsilon t$. Claim (1) follows in the same way as in the proof of Theorem 3.2. For (2), let I be any vertex cover of size at most t, and consider the walk from root to leaf on the search tree, where at a node with label (G', n', t', v', S'), we turn right if $v' \in I$, and turn left otherwise. Then it is not hard to verify that at each step it holds that $S' \subseteq I$. Consider the corresponding leaf, and let (G', n', t', v', S') be the original label of the leaf. If G' has no edges then $S' \subseteq I$ and we're done. Otherwise, the set $I \setminus S'$ forms a vertex cover of G' of size at most $t' \le \epsilon t$, and therefore, the 2-approximation algorithm returns a vertex cover S'' of G' of size at most $2t' \le 2\epsilon t$. We conclude that $|S' \setminus I| \le |S''| \le 2\epsilon t$, as required.

Running time. Let $T(n, t, \epsilon t)$ be the size of the largest search tree that is truncated according to parameter ϵt, among the search trees of all graphs with n vertices that are t-edge-neighborhood graphs. Observe that the search tree can be constructed in time $\mathrm{poly}(T(n, t, \epsilon t), n)$, and therefore, it is enough to prove that $T(n, t, \epsilon t) = t^{O(\log(1/\epsilon))}$.

As in the proof of Theorem 3.2 we observe that for every node in the search tree with label (G', n', t', v', S'), it holds that G' is a t'-edge-neighborhood graph, so v' has degree at least $\lceil (t'+1)/2 \rceil$ in G'. Therefore, from any path from root to leaf, the number of left steps can be at most $\lceil \log(1/\epsilon) \rceil + 1$. In addition, the total number of steps in such a path is at most ϵt, and therefore the total number of such paths is $\binom{\epsilon t}{\lceil \log(1/\epsilon) \rceil + 1} = t^{O(\log(1/\epsilon))}$, and $T(n, t) \le (t+1) \cdot t^{O(\log(1/\epsilon))} = t^{O(\log(1/\epsilon))}$. This completes the proof of the theorem.

$\square$

4 Comparison-Based Codes

In this section we present our results regarding comparison-based codes, formally defined as follows.

Definition 4.1 (Comparison-based codes). *A code $\mathcal{C} \subseteq [q]^n$ is a* comparison-based *code, if for every $1 \le i < j \le n$ there exist functions $f_{i,j}, f_{j,i} : [q] \to [q]$ such that the (i, j)-th conflict function $G_{i,j}$ is given by $G_{i,j}(\sigma, \tau) = \mathsf{NEQ}(f_{i,j}(\sigma), f_{j,i}(\tau))$, where $\mathsf{NEQ}(x, y)$ is the not-equal function, that returns 1 if $x \neq y$, and 0 if $x = y$.*

The rest of this section is organised as follows. In Section 4.1 we present a lower bound for comparison-based conflict checkable codes that satisfy local-to-global consistency. In Section 4.2 we discuss linear codes, and show that they are comparison-based codes. In Section 4.3 we present a general framework for constructing comparison-based robust conflict decodable codes, based on linear MDS codes.

4.1 Lower Bound on Comparison-Based Codes

In this section we prove Theorem 1.8, that we repeat here.

Theorem 4.2 (Theorem 1.8 restated)**.** *Let $\mathcal{C}$ be an $(n,k,d)_q$ comparison-based conflict checkable code with $1 < d < n$, and assume that it satisfies local-to-global consistency. Then $k \le \frac{n-d+2}{2}$.*

We prove Theorem 4.2 in Section 4.1.1. Since the proof of Theorem 4.2 is somewhat technical, we first consider the toy version where $n = 3$ and $d = 2$, that conveys the main ideas of the proof.

Toy version. Consider the case where $n = 3$ and $d = 2$, so our goal is to prove that $k \le 3/2$. Let (X_1, X_2, X_3) be a uniformly distributed codeword. In the first step, we bound the Shannon's entropy of $f_{1,2}(X_1)$. Consider the entropy of (X_1, X_2, X_3), and observe that

$$k \cdot \log q = H(X_1, X_2, X_3) = H(X_1, X_2) \le H(X_1) + H(X_2) - H(f_{1,2}(X_1)).$$

The second equality follows since the code has distance $d = 2$ and so every $n - d + 1 = 2$ entries fully determine the codeword. The inequality follows by noting that the random variable $f_{1,2}(X_1)$ is equal to $f_{2,1}(X_2)$ and can therefore be (deterministically) derived both from X_1 and from X_2. Therefore $H(f_{1,2}(X_1)) \le H(X_1) + H(X_2) - k \cdot \log q$. A similar argument shows that $H(f_{1,3}(X_1)) \le H(X_1) + H(X_3) - k \cdot \log q$.

In the second step, we analyse the joint entropy $H(f_{1,2}(X_1), f_{1,3}(X_1))$. Since the code has distance $d = n - 1 = 2$, for every choice (σ_2, σ_3) in the support of (X_2, X_3), there exists a unique σ_1 such that $(\sigma_1, \sigma_2, \sigma_3)$ is a codeword. Since the code is conflict checkable, this means that there exists a unique σ_1 that satisfies $f_{1,2}(\sigma_1) = f_{2,1}(\sigma_2)$ and $f_{1,3}(\sigma_1) = f_{3,1}(\sigma_3)$. We conclude that $f_{1,2}(X_1)$ and $f_{1,3}(X_1)$ fully determine X_1, and therefore $H(f_{1,2}(X_1), f_{1,3}(X_1)) = H(X_1)$.

Finally, in the third step, we analyse the mutual information $I(f_{1,2}(X_1); f_{1,3}(X_1))$:

$$\begin{aligned}0 \le I(f_{1,2}(X_1); f_{1,3}(X_1)) &= H(f_{1,2}(X_1)) + H(f_{1,3}(X_1)) - H(f_{1,2}(X_1), f_{1,3}(X_1)) \\ &\le \Big(H(X_1) + H(X_2) - k \cdot \log q\Big) + \Big(H(X_1) + H(X_3) - k \cdot \log q\Big) - H(X_1),\end{aligned}$$

where the last inequality is based on the bounds from the first and second step. Since $H(X_1), H(X_2), H(X_3) \le \log q$, we obtain that $k \le 3/2$, as required.

The proof of the general case follows the same lines, but is somewhat more technical. First, we can no longer take a single index as a pivot (the index $i = 1$ in the toy version). For example, in the first step we need to obtain a bound on $H(f_{i,j}(X_i))$ for every pair of indices i, j. Moreover, we can no longer restrict ourselves to a fixed set of $n - d + 1$ indices, but rather, the bound on $H(f_{i,j}(X_i))$ is obtained by averaging over the bounds obtained from every possible set of $n - d + 1$ indices. The third step is also generalized in similar ways. Finally, in order to perform the second step when $d < n - 1$, we explicitly use the assumption that the code $\mathcal{C}$ satisfies local-to-global consistency. (In the toy version this property follows implicitly since $d = n - 1$.)

4.1.1 Proof of Theorem 4.2

Let $\mathcal{C}$ be an $(n,k,d)_q$ code with $1 < d < n$, so necessarily $n \ge 3$. We assume that $\mathcal{C}$ is conflict checkable and satisfies local-to-global consistency, and that it is comparison-based, i.e., for every

$1 \le i < j \le n$ there exist functions $f_{i,j}, f_{j,i} : [q] \to [q]$ such that $G_{i,j}(\sigma, \tau) = \mathsf{NEQ}(f_{i,j}(\sigma), f_{j,i}(\tau))$. Our goal is to prove that the dimension k of the code is at most $(n-d+2)/2$.

Let $(X_1, \ldots, X_n)$ be a uniformly distributed codeword from $\mathcal{C}$. We denote by Π the set of all $n!$ permutations of the vector $(1, \ldots, n)$, and for every pair of distinct indices $i_1, i_2 \in [n]$, we let $S(i_1, i_2)$ be the set of all vectors of length $n-2$ that contain distinct elements from $\{1, \ldots, n\} \setminus \{i_1, i_2\}$. The proof continues in three steps. First, we prove the following upper bound on the entropy of $f_{i,j}(X_i)$ (that corresponds to the first step in the toy version).

Claim 4.3. *For every pair of distinct indices $i_1, i_2 \in [n]$ it holds that*

$$\begin{aligned}
H(f_{i_1,i_2}(X_{i_1})) \le\ & H(X_{i_1}) + H(X_{i_2}) + \frac{1}{(n-2)!} \sum_{(i_3,\ldots,i_n)\in S(i_1,i_2)} (H(X_{i_3}) + \ldots + H(X_{i_{n-d+1}})) \\
& - \frac{1}{(n-2)!} \sum_{(i_3,\ldots,i_n)\in S(i_1,i_2)} (H(f_{i_1,i_3}(X_{i_1}), f_{i_2,i_3}(X_{i_2})) + \ldots + H(f_{i_1,i_{n-d+1}}(X_{i_1}), \ldots, f_{i_{n-d},i_{n-d+1}}(X_{i_{n-d}}))) \\
& - k \log q,
\end{aligned}$$

where the sum in the second row is 0 if $d = n-1$.

Then, we prove that X_i is fully determined by any $n-d+1$ evaluations $f_{i,j_1}(X_i), \ldots, f_{i,j_{n-d+1}}(X_i)$, as stated in the following claim (that corresponds to the second step in the toy version).

Claim 4.4. *For every permutation $(i_1, \ldots, i_n) \in \Pi$ it holds that*

$$H(X_{i_1}) = H(f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_{n-d+2}}(X_{i_1})),$$

where we observe that, since $d > 1$, then $n-d+2 \le n$ and the RHS of the equation is well defined.

The proofs of Claim 4.3 and Claim 4.4 are deferred to Section 4.1.2 and Section 4.1.3, respectively. Let us continue with the proof of Theorem 4.2 given Claim 4.3 and Claim 4.4. For every permutation $(i_1, \ldots, i_n) \in \Pi$ and $2 \le t \le n-d+1$, since the mutual information is non-negative, we have[5]

$$\begin{aligned}
0 &\le I(f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_t}(X_{i_1}); f_{i_1,i_{t+1}}(X_{i_1}), \ldots, f_{i_1,i_{n-d+2}}(X_{i_1})) \\
&= H(f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_t}(X_{i_1})) + H(f_{i_1,i_{t+1}}(X_{i_1}), \ldots, f_{i_1,i_{n-d+2}}(X_{i_1})) - H(f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_{n-d+2}}(X_{i_1})) \\
&= H(f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_t}(X_{i_1})) - H(X_{i_1}) + H(f_{i_1,i_{t+1}}(X_{i_1}), \ldots, f_{i_1,i_{n-d+2}}(X_{i_1})),
\end{aligned}$$

where in the last equality we used Claim 4.4 and reordered the terms. Summing over all $(i_1, \ldots, i_n) \in \Pi$ we obtain

$$\begin{aligned}
0 &\le \sum_{(i_1,\ldots,i_n)} \big(H(f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_t}(X_{i_1})) - H(X_{i_1}) + H(f_{i_1,i_{t+1}}(X_{i_1}), \ldots, f_{i_1,i_{n-d+2}}(X_{i_1}))\big) \\
&= (n-t)! \cdot \sum_{(i_1,\ldots,i_t)} H(f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_t}(X_{i_1})) - (n-1)! \sum_{i=1}^{n} H(X_i) \\
&\quad + (d+t-3)! \cdot \sum_{(i_1,\ldots,i_{n-d-t+3})} H(f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_{n-d-t+3}}(X_{i_1})).
\end{aligned}$$

[5]Observe that $n-d+1 \ge n-(n-1)+1 = 2$, so there is at least one possible value for t, and that $n-d+2 \le n-2+2 = n$.

For every $2 \le t \le n-d+1$ define

$$f(t) := (n-t)! \cdot \sum_{(i_1,\ldots,i_t)} H(f_{i_1,i_2}(X_{i_1}),\ldots,f_{i_1,i_t}(X_{i_1})) - (n-1)! \sum_{i=1}^{n} H(X_i)$$
$$+ (d+t-3)! \cdot \sum_{(i_1,\ldots,i_{n-d-t+3})} H(f_{i_1,i_2}(X_{i_1}),\ldots,f_{i_1,i_{n-d-t+3}}(X_{i_1})),$$

observe that $f(t)$ is just the sum of $I(f_{i_1,i_2}(X_{i_1}),\ldots,f_{i_1,i_t}(X_{i_1}); f_{i_1,i_{t+1}}(X_{i_1}),\ldots,f_{i_1,i_{n-d+2}}(X_{i_1}))$ over all $(i_1,\ldots,i_n) \in \Pi$, and that we proved that $f(t) \ge 0$. The rest of the proof is ordered as follows. In Claim 4.5 we prove Theorem 4.2 for the special case of $d = n-1$. This proof is short and similar to the proof of the toy version. Then, in Claim 4.6 we prove Theorem 4.2 for $d < n-1$, for which the proof is more involved.

Claim 4.5. *Assume that $d = n-1$. Then $k \le (n-d+2)/2$.*

Proof. For $d = n-1$ there is a single choice for $2 \le t \le n-d+1$ which is $t = 2$. Recall that in the toy version we achieved the bound $k \le (n-d+2)/2$ by bounding the term $I(f_{1,2}(X_1); f_{1,3}(X_1))$. Proving Claim 4.5 follows similar lines, where we bound the term $f(2)$, which is just the sum of $I(f_{i_1,i_2}(X_{i_1}); f_{i_1,i_3}(X_{i_1}))$ over all $(i_1,\ldots,i_n) \in \Pi$. Observe that $f(2)$ is bounded by

$$\begin{aligned}
f(2) &= (n-2)! \cdot \sum_{(i_1,i_2)} H(f_{i_1,i_2}(X_{i_1})) - (n-1)! \sum_{i=1}^{n} H(X_i) \\
&\quad + (n-2)! \cdot \sum_{(i_1,i_2)} H(f_{i_1,i_2}(X_{i_1})) \\
&= 2(n-2)! \cdot \sum_{(i_1,i_2)} H(f_{i_1,i_2}(X_{i_1})) - (n-1)! \sum_{i=1}^{n} H(X_i) \\
&\le 2(n-2)! \sum_{(i_1,i_2)} (H(X_{i_1}) + H(X_{i_2})) - 2(n-2)! \cdot n \cdot (n-1) k \log q - (n-1)! \sum_{i=1}^{n} H(X_i) \\
&= 3(n-1)! \cdot \sum_{i=1}^{n} H(X_i) - 2n! \cdot k \log q,
\end{aligned}$$

where the inequality follows from Claim 4.3 for the special case of $d = n-1$. Since $f(2) \ge 0$ and $H(X_i) \le \log q$ for every $i \in [n]$, it holds that $k \le 3/2 = (n-d+2)/2$, as required. This concludes the proof of Claim 4.5. ☐

Claim 4.6. *Assume that $d < n-1$. Then $k \le (n-d+2)/2$.*

Proof. First, we bound the term $f(2)$.

$$f(2) = (n-2)! \cdot \sum_{(i_1,i_2)} H(f_{i_1,i_2}(X_{i_1})) - (n-1)! \sum_{i=1}^{n} H(X_i)$$
$$+ (d-1)! \cdot \sum_{(i_1,\ldots,i_{n-d+1})} H(f_{i_1,i_2}(X_{i_1}),\ldots,f_{i_1,i_{n-d+1}}(X_{i_1}))$$

$$
\begin{aligned}
&\le (n-2)! \sum_{(i_1,i_2)} (H(X_{i_1}) + H(X_{i_2})) + \sum_{(i_1,\dots,i_n)} (H(X_{i_3}) + \ldots + H(X_{i_{n-d+1}})) \\
&\quad - \sum_{(i_1,\dots,i_n)} (H(f_{i_1,i_3}(X_{i_1}), f_{i_2,i_3}(X_{i_2})) + \ldots + H(f_{i_1,i_{n-d+1}}(X_{i_1}), \ldots, f_{i_{n-d},i_{n-d+1}}(X_{i_{n-d}}))) \\
&\quad - (n-2)! \cdot n \cdot (n-1) k \log q \\
&\quad - (n-1)! \sum_{i=1}^{n} H(X_i) + (d-1)! \cdot \sum_{(i_1,\dots,i_{n-d+1})} H(f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_{n-d+1}}(X_{i_1})),
\end{aligned}
$$

where the inequality follows from Claim 4.6, and we observe that no summation is empty, since $d < n-1$. Therefore,

$$
\begin{aligned}
f(2) &\le 2 \cdot (n-1)! \sum_{i=1}^{n} H(X_i) + (n-d-1) \cdot (n-1)! \sum_{i=1}^{n} H(X_i) \\
&\quad - (n-3)! \sum_{(i_1,\dots,i_3)} (H(f_{i_1,i_3}(X_{i_1}), f_{i_2,i_3}(X_{i_2})) - (n-4)! \sum_{(i_1,\dots,i_4)} (H(f_{i_1,i_4}(X_{i_1}), f_{i_2,i_4}(X_{i_2}), f_{i_3,i_4}(X_{i_3})) \\
&\quad - \ldots - (d-1)! \sum_{(i_1,\dots,i_{n-d+1})} H(f_{i_1,i_{n-d+1}}(X_{i_1}), \ldots, f_{i_{n-d},i_{n-d+1}}(X_{i_{n-d}}))) \\
&\quad - n! \cdot k \log q \\
&\quad - (n-1)! \sum_{i=1}^{n} H(X_i) + (d-1)! \cdot \sum_{(i_1,\dots,i_{n-d+1})} H(f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_{n-d+1}}(X_{i_1})) \\
&= (n-d) \cdot (n-1)! \sum_{i=1}^{n} H(X_i) \\
&\quad - (n-3)! \sum_{(i_1,\dots,i_3)} (H(f_{i_1,i_2}(X_{i_1}), f_{i_1,i_3}(X_{i_1})) - (n-4)! \sum_{(i_1,\dots,i_4)} (H(f_{i_1,i_2}(X_{i_1}), f_{i_1,i_3}(X_{i_1}), f_{i_1,i_4}(X_{i_1})) \\
&\quad - \ldots - d! \sum_{(i_1,\dots,i_{n-d})} H(f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_{n-d}}(X_{i_1}))) \\
&\quad - n! \cdot k \log q, \qquad (8)
\end{aligned}
$$

where in the last equality we simply changed the name of the indices, and used the fact that $f_{i,j}(X_i) = f_{j,i}(X_j)$ for every $i, j \in [n]$ with probability 1. To complete the proof we split into cases according to the parity of $(n-d+2)$.

First case: $(n-d+2)$ **is even.** Assume that $(n-d+2)$ is even. Consider the sum $0 \le f(2) + \sum_{t=3}^{(n-d+2)/2} f(t)$, observe that every $f(t)$ for $t > 2$ cancels

$$
(n-t)! \cdot \sum_{(i_1,\dots,i_t)} H(f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_t}(X_{i_1})) \quad \text{and} \quad (d+t-3)! \cdot \sum_{(i_1,\dots,i_{n-d-t+3})} H(f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_{n-d-t+3}}(X_{i_1})).
$$

in the bound of $f(2)$ in Equation 8, and therefore,[6]

$$n! \cdot k \log q \leq (n-d) \cdot (n-1)! \cdot \sum_{i=1}^{n} H(X_i) - (\frac{n-d-2}{2}) \cdot (n-1)! \cdot \sum_{i=1}^{n} H(X_i)$$
$$\leq \frac{n-d+2}{2} \cdot n! \log q$$

so $k \leq \frac{n-d+2}{2}$, as required.

Second case: $(n-d+2)$ **is odd.** Assume that $(n-d+2)$ is odd. Consider the sum $0 \leq f(2) + \sum_{t=3}^{(n-d+1)/2} f(t)$, observe that every $f(t)$ for $t > 2$ cancels

$$(n-t)! \cdot \sum_{(i_1,\ldots,i_t)} H(f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_t}(X_{i_1})) \quad \text{and} \quad (d+t-3)! \cdot \sum_{i_1,\ldots,i_{n-d-t+3}} H(f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_{n-d-t+3}}(X_{i_1})).$$

in the bound of $f(2)$ in Equation 8, and therefore,[7]

$$n! \cdot k \log q \leq (n-d) \cdot (n-1)! \cdot \sum_{i=1}^{n} H(X_i) - \frac{n-d-3}{2} \cdot (n-1)! \cdot \sum_{i=1}^{n} H(X_i)$$
$$-(\frac{n+d-3}{2})! \cdot \sum_{i_1,\ldots,i_{\frac{n-d+3}{2}}} H(f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_{\frac{n-d+3}{2}}} H(X_{i_1})).$$

For $t = \frac{n-d+3}{2}$ we have $(n-t)! = (d+t-3)!$, and since $f(t) \geq 0$ we obtain

$$(n-1)! \sum_{i=1}^{n} H(X_i) \leq 2(\frac{n+d-3}{2})! \sum_{i_1,\ldots,i_{\frac{n-d+3}{2}}} H(f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_{\frac{n-d+3}{2}}} H(X_{i_1})).$$

We conclude that

$$n!k \log q \leq \big((n-d) \cdot (n-1)! \cdot n - \frac{n-d-3}{2} \cdot (n-1)! \cdot n - \frac{1}{2}(n-1)! \cdot n\big) \log q = \frac{n-d+2}{2} \cdot n! \cdot \log q,$$

so $k \leq \frac{n-d+2}{2}$, as required. This concludes the proof of Claim 4.6. □

The proof of Theorem 4.2 now follows from Claim 4.5 and Claim 4.6.

4.1.2 Proof of Claim 4.3

In this Section we prove Claim 4.3. First we observe that, since it is possible to recover from $d-1$ erasures, for every $n-d+1$ distinct indices $i_1, \ldots, i_{n-d+1} \in \{1, \ldots, n\}$ it holds that

$$k \log q = H(X_1, \ldots, X_n) = H(X_{i_1}, \ldots, X_{i_{n-d+1}}).$$

[6] For the special case of $d = n-2$ the sum $\sum_{t=3}^{(n-d+2)/2} f(t)$ is empty. However, in this case the following follows directly from the bound on $f(2)$.

[7] For the special case of $d = n-3$ the sum $\sum_{t=3}^{(n-d+1)/2} f(t)$ is empty. However, in this case the following follows directly from the bound on $f(2)$.

Therefore, for every permutation $(i_1, \ldots, i_n) \in \Pi$ it holds that

$$\begin{aligned}
k \log q &= H(X_{i_1}, \ldots, X_{i_{n-d+1}}) \\
&= H(X_{i_1}) + H(X_{i_2} \mid X_{i_1}) + H(X_{i_3} \mid X_{i_1}, X_{i_2}) \ldots + H(X_{i_{n-d+1}} \mid X_{i_1}, \ldots, X_{i_{n-d}}) \\
&\leq H(X_{i_1}) + H(X_{i_2} \mid f_{i_1,i_2}(X_{i_1})) \\
&\quad + H(X_{i_3} \mid f_{i_1,i_3}(X_{i_1}), f_{i_2,i_3}(X_{i_2})) + \ldots + H(X_{i_{n-d+1}} \mid f_{i_1,i_{n-d+1}}(X_{i_1}), \ldots, f_{i_{n-d},i_{n-d+1}}(X_{i_{n-d}})) \\
&= H(X_{i_1}) + H(X_{i_2}) + H(X_{i_3}) + \ldots + H(X_{i_{n-d+1}}) \\
&\quad - H(f_{i_1,i_2}(X_{i_1})) - H(f_{i_1,i_3}(X_{i_1}), f_{i_2,i_3}(X_{i_2})) - \ldots - H(f_{i_1,i_{n-d+1}}(X_{i_1}), \ldots, f_{i_{n-d},i_{n-d+1}}(X_{i_{n-d}}))
\end{aligned}$$

where in the last equality we used the chain rule and the fact that $f_{i,j}(X_i) = f_{j,i}(X_j)$ with probability 1, for every $i, j \in [n]$. Note that the sum in the last row is not empty since $d < n$. Hence,

$$\begin{aligned}
H(f_{i_1,i_2}(X_{i_1})) \leq\ & H(X_{i_1}) + H(X_{i_2}) + H(X_{i_3}) + \ldots + H(X_{i_{n-d+1}}) \\
& - H(f_{i_1,i_3}(X_{i_1}), f_{i_2,i_3}(X_{i_2})) - \ldots - H(f_{i_1,i_{n-d+1}}(X_{i_1}), \ldots, f_{i_{n-d},i_{n-d+1}}(X_{i_{n-d}})) \\
& - k \log q, \qquad (9)
\end{aligned}$$

where the sum in the second row is 0 if $d = n - 1$.

Consider now any pair of distinct indices $i_1, i_2 \in [n]$, and recall that $S(i_1, i_2)$ is the set of all vectors of length $n-2$ that contain distinct elements from $\{1, \ldots, n\} \backslash \{i_1, i_2\}$, so $|S(i_1, i_2)| = (n-2)!$. As Equation 9 holds for every choice of $(i_3, \ldots, i_n) \in S(i_1, i_2)$, we can take the average of these inequalities and obtain

$$\begin{aligned}
H(f_{i_1,i_2}(X_{i_1})) \leq\ & H(X_{i_1}) + H(X_{i_2}) + \frac{1}{(n-2)!} \sum_{(i_3,\ldots,i_n) \in S(i_1,i_2)} (H(X_{i_3}) + \ldots + H(X_{i_{n-d+1}})) \\
& - \frac{1}{(n-2)!} \sum_{(i_3,\ldots,i_n) \in S(i_1,i_2)} (H(f_{i_1,i_3}(X_{i_1}), f_{i_2,i_3}(X_{i_2})) + \ldots + H(f_{i_1,i_{n-d+1}}(X_{i_1}), \ldots, f_{i_{n-d},i_{n-d+1}}(X_{i_{n-d}}))) \\
& - k \log q.
\end{aligned}$$

This completes the proof of the claim.

4.1.3 Proof of Claim 4.4

In this section we prove Claim 4.4. We argue that for every permutation $(i_1, \ldots, i_n) \in \Pi$ it holds that

$$H(X_{i_1}) = H(f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_{n-d+2}}(X_{i_1})).$$

To do so, it is enough to prove that $H(X_{i_1} \mid f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_{n-d+2}}(X_{i_1})) = 0$. Assume towards contradiction that $H(X_{i_1} \mid f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_{n-d+2}}(X_{i_1})) > 0$, so there exists a codeword $\mathbf{c} \in \mathcal{C}$, $\mathbf{c} = (\sigma_1, \ldots, \sigma_n)$, such that $H(X_{i_1} \mid f_{i_1,i_2}(\sigma_{i_1}), \ldots, f_{i_1,i_{n-d+2}}(\sigma_{i_1})) > 0$. That is, there exists $\tau_{i_1} \neq \sigma_{i_1}$ that satisfies $f_{i_1,i_2}(\sigma_{i_1}) = f_{i_1,i_2}(\tau_{i_1}), \ldots, f_{i_1,i_{n-d+2}}(\sigma_{i_1}) = f_{i_1,i_{n-d+2}}(\tau_{i_1})$. Consider the sub-vector $(\tau_{i_1}, \sigma_{i_2}, \ldots, \sigma_{i_{n-d+2}}) \in [q]^{n-d+2}$ and observe that every two entries are pairwise consistent. Since $\mathcal{C}$ satisfies local-to-global consistency, this vector fully defines a codeword $\mathbf{c}' \in \mathcal{C}$ such that $\mathbf{c}' \neq \mathbf{c}$ since $\mathbf{c}'[i_1] = \tau_{i_1} \neq \sigma_{i_1} = \mathbf{c}[i_1]$. However, the distance of $\mathbf{c}$ and $\mathbf{c}'$ is at most $(d-2) + 1 = d - 1$, in contradiction. Therefore, $H(X_{i_1}) = H(f_{i_1,i_2}(X_{i_1}), \ldots, f_{i_1,i_{n-d+2}}(X_{i_1}))$, as required. This completes the proof of the claim.

4.2 Linear Conflict checkable Codes are Comparison-Based Codes

Let $\mathbb{F}$ be a finite field, and let $q = |\mathbb{F}|^\ell$ for some positive integer ℓ. We say that an $(n,k,d)_q$ code $\mathcal{C}$ is a *linear* code over $\mathbb{F}$, and refer to it as an $[n,k,d]_q$ code, if $\mathcal{C} \subseteq (\mathbb{F}^\ell)^n$, and there exist n matrices $\mathcal{G}_1,\ldots,\mathcal{G}_n \in \mathbb{F}^{\ell\times(\ell\cdot k)}$ such that

- It holds that $\dim(\text{Row-Span}(\mathcal{G}_1,\ldots,\mathcal{G}_n)) = \ell\cdot k$. (That is, the $(\ell\cdot n)\times(\ell\cdot k)$ matrix $\mathcal{G}$ obtained by putting the matrices $\mathcal{G}_1,\ldots,\mathcal{G}_n$ one on top of the other, has full rank.)
- For every codeword $\mathbf{c} = (\mathbf{c}[1],\ldots,\mathbf{c}[n]) \in \mathcal{C}$ there exists an information word $\mathbf{m} \in \mathbb{F}^{\ell\cdot k}$ such that $\mathbf{c}[i] = \mathcal{G}_i \cdot \mathbf{m}$ for every $i \in [n]$.

It is not hard to see that for every codeword $\mathbf{c} \in \mathcal{C}$ there exists a *unique* information word $\mathbf{m} \in \mathbb{F}^{\ell\cdot k}$ such that $\mathbf{c}[i] = \mathcal{G}_i \cdot \mathbf{m}$ for every $i \in [n]$. Note that the distance of the code is d if and only $d = \min_{\mathbf{c}\in\mathcal{C}} |\{i \in [n] \mid \mathbf{c}[i] \neq \vec{0}\}|$. We also note that k is not necessarily an integer. We continue by proving that every linear conflict checkable code is a comparison-based code.

Theorem 4.7. *Let $\mathbb{F}$ be a finite field, and let $q = |\mathbb{F}|^\ell$ for some positive integer ℓ. Let $\mathcal{C}$ be an $[n,k,d]_q$ code over $\mathbb{F}$. Then $\mathcal{C}$ is a comparison-based code.*

Combining Theorem 4.7 with Theorem 4.2, we immediately obtain the following corollary.

Corollary 4.8. *Let $\mathbb{F}$ be a finite field, and let $q = |\mathbb{F}|^\ell$ for some positive integer ℓ. Let $\mathcal{C}$ be an $[n,k,d]_q$ conflict checkable code that satisfies local-to-global consistency. Then $k \leq (n-d+2)/2$.*

We continue with the proof of Theorem 4.7.

Proof. We show that for every $1 \leq i < j \leq n$, the conflict function $G_{i,j}$ is of the form $G_{i,j}(\mathbf{u}_i,\mathbf{u}_j) = \mathsf{NEQ}(f_{i,j}(\mathbf{u}_i), f_{j,i}(\mathbf{u}_j))$, where in our context $\mathbf{u}_i,\mathbf{u}_j \in \mathbb{F}^\ell$. Let $\mathcal{G}_i$ and $\mathcal{G}_j$ be the corresponding matrices, and let $\mathcal{G}$ be the $2\ell \times \ell\cdot k$ matrix defined by

$$\mathcal{G} := \left[\frac{\mathcal{G}_i}{\mathcal{G}_j}\right], \quad \text{and let} \quad \mathbf{u} := \left[\frac{\mathbf{u}_i}{\mathbf{u}_j}\right].$$

Let $\text{Im}(\mathcal{G})$ be the image of $\mathcal{G}$, let $V := \text{Im}(\mathcal{G})^\perp$ be the orthogonal complement of the image, and let $\{\mathbf{v}_1,\ldots,\mathbf{v}_r\}$ be a basis of V. Observe that $G_{i,j}(\mathbf{u}_i,\mathbf{u}_j) = 0$ if and only if $\mathbf{u} \in \text{Im}(\mathcal{G})$, if and only if $\mathbf{u}\cdot\mathbf{v}_i = 0$ for every $i \in [r]$, and that this occurs if and only if $\mathbf{u}_i \cdot \mathsf{pre}(\mathbf{v}_z) = -\mathbf{u}_j \cdot \mathsf{suff}(\mathbf{v}_z)$ for all $z \in [r]$, where $\mathsf{pre}(\mathbf{v}_z)$ is the length-ℓ prefix of $\mathbf{v}_z$, and $\mathsf{suff}(\mathbf{v}_z)$ is the length-ℓ suffix of $\mathbf{v}_z$. Therefore, we define $f_{i,j}(\mathbf{u}_i) := (\mathbf{u}_i \cdot \mathsf{pre}(\mathbf{v}_z))_{z\in[r]}$ and $f_{j,i}(\mathbf{u}_j) := (-\mathbf{u}_j \cdot \mathsf{suff}(\mathbf{v}_z))_{z\in[r]}$, and indeed it holds that $G_{i,j}(\mathbf{u}_i,\mathbf{u}_j) = \mathsf{NEQ}(f_{i,j}(\mathbf{u}_i), f_{j,i}(\mathbf{u}_j))$. This concludes the proof of the theorem. $\square$

4.3 Comparison-Based Codes from any Linear MDS Code

In this section we present our construction of comparison-based codes based on any linear MDS code. The section is organised as follows in Section 4.3.1 we recall the definition and some basic properties of *multilinear forms*, that will be used in the construction. In Section 4.3.2 we present the basic construction, and in Section 4.3.3 we show that it can be used to construct optimal comparison-based codes, albeit with inefficient conflict-decoder. In Section 4.3.4 we show that in the special case of $n \geq 4t$ we can obtain a code with *polynomial-time* conflict-decoder, and in Section 4.3.5 we present an asymptotically-good code (i.e., a non-optimal code with constant rate and constant relative distance) with *quasipolynomial-time* conflict decoder. Finally, in Section 4.3.6 we show that our codes can be used as a threshold secret sharing scheme.

4.3.1 Multilinear Forms

We recall the notion of multilinear forms from linear algebra. For more information, see, e.g., [RAG05]. Let V be a vector space over a field $\mathbb{F}$. A function $F : V^m \to \mathbb{F}$ is a *multilinear form* (or *m-form*) if it is linear in each coordinate separately, i.e., if

$$\begin{aligned} F(\mathbf{v}_1,\ldots,\mathbf{v}_{i-1},\alpha\mathbf{v}_i+\beta\mathbf{v}'_i,\mathbf{v}_{i+1},\ldots,\mathbf{v}_m) = {} & \alpha F(\mathbf{v}_1,\ldots,\mathbf{v}_{i-1},\mathbf{v}_i,\mathbf{v}_{i+1},\ldots,\mathbf{v}_m) \\ & + \beta F(\mathbf{v}_1,\ldots,\mathbf{v}_{i-1},\mathbf{v}'_i,\mathbf{v}_{i+1},\ldots,\mathbf{v}_m), \end{aligned}$$

for every $i \in [m]$, $\mathbf{v}_1,\ldots,\mathbf{v}_m,\mathbf{v}'_i \in V$ and $\alpha,\beta \in \mathbb{F}$. The function F is a *symmetric* multilinear form if for every permutation π of $[m]$ it holds that

$$F(\mathbf{v}_1,\ldots,\mathbf{v}_m) = F(\mathbf{v}_{\pi(1)},\ldots,\mathbf{v}_{\pi(m)}).$$

Observe that the set of all symmetric m-forms is a vector space over $\mathbb{F}$. The following lemmas will be useful for the analysis of our construction. Proofs appear in Appendix B.

Lemma 4.9. *Let $m,t \geq 1$ and $\ell \geq t$ be integers, let V be a vector space of dimension t over a field $\mathbb{F}$, let $\mathbf{v}_1,\ldots,\mathbf{v}_\ell \in V$ be vectors spanning V, and let $F_1,\ldots,F_\ell$ be symmetric $(m-1)$-forms that satisfy $F_i(\mathbf{v}_j,\cdot,\ldots,\cdot) = F_j(\mathbf{v}_i,\cdot,\ldots,\cdot)$ for all $i \neq j$. Then there exists a unique m-form F that satisfies*

$$F(\mathbf{v}_i,\cdot,\ldots,\cdot) = F_i(\cdot,\ldots,\cdot),$$

for all $i \in [\ell]$. In addition, F is symmetric.

Lemma 4.10. *Let $m,t \geq 1$ be integers, let V be a vector space of dimension t over a finite field $\mathbb{F}$, and let $0 \leq \ell \leq t$ be an integer. Let $\mathbf{v}_1,\ldots,\mathbf{v}_\ell \in V$ be linearly independent vectors in V, and let $F_1,\ldots,F_\ell$ be symmetric $(m-1)$-forms that satisfy $F_i(\mathbf{v}_j,\cdot,\ldots,\cdot) = F_j(\mathbf{v}_i,\cdot,\ldots,\cdot)$ for all $i \neq j$. Then the number of symmetric m-forms F that satisfy $F(\mathbf{v}_i,\cdot,\ldots,\cdot) = F_i(\cdot,\ldots,\cdot)$ for all $i \in [\ell]$ is exactly*

$$|\mathbb{F}|^L, \quad \textit{where} \quad L = \binom{m+(t-\ell-1)}{m}.$$

4.3.2 The Construction

In this section we present our construction of comparison-based robust conflict decodable codes, and prove some basic properties. Let $\mathcal{G} \in \mathbb{F}^{N\times K}$ be the generator matrix of a linear $[N,K,D = N-K+1]_{|\mathbb{F}|}$ MDS-code $\mathcal{S}$, and denote by $\mathcal{G}_i$ the i-th row of $\mathcal{G}$. Let $m \geq 1$ be an integer and consider the following construction.

Construction 4.11. *Define the length-N code*

$$\mathcal{C} := \{(F(\mathcal{G}_1,\cdot,\ldots,\cdot),\ldots,F(\mathcal{G}_N,\cdot,\ldots,\cdot) \mid F : (\mathbb{F}^K)^m \to \mathbb{F} \textit{ is a symmetric } m\textit{-form}\}.$$

That is, for every F we define a codeword whose ith coordinate is the symmetric $(m-1)$-form that is obtained from F by restricting its first input to the ith row of $\mathcal{G}$.

Theorem 4.12. *The code $\mathcal{C}$ in Construction 4.11 is an $[n,k,d]_q$ code, for code-length $n = N$, alphabet size $q = |\mathbb{F}|^{\binom{m+K-2}{m-1}}$, dimension $k = 1 + \frac{K-1}{m}$ and distance $d = D$. In addition, for $m \geq 2$ the code is a conflict checkable code that satisfies local-to-global consistency, and for every $1 \leq i < j \leq n$ the conflict function $G_{i,j}$ is given by $G_{i,j}(F_i,F_j) = \mathsf{NEQ}(F_i(\mathcal{G}_j,\cdot,\ldots,\cdot),F_j(\mathcal{G}_i,\cdot,\ldots,\cdot))$.*

The following corollary follows immediately from Lemma 1.11.

Corollary 4.13. *In the settings of Theorem 4.12, the code $\mathcal{C}$ is a comparison-based t-robust conflict decodable code for $t = \lfloor (d-1)/2 \rfloor$.*

We continue with the proof of Theorem 4.12.

Proof of Theorem 4.12. Observe that $\mathcal{C}$ is indeed a code with length n. Every symbol is a symmetric $(m-1)$-form, and by Lemma 4.10 the number of symmetric $(m-1)$-forms is $|\mathbb{F}|^{\binom{(m-1)+K-1}{m-1}}$. Therefore, the size of the alphabet is $q = |\mathbb{F}|^{\binom{m+K-2}{m-1}}$. We continue with the analysis of the distance and the dimension.

Distance. Consider two distinct symmetric m-forms F and F', and let $\mathbf{c}$ and $\mathbf{c}'$ be the corresponding codewords. We prove by induction on m that the distance of $\mathbf{c}$ and $\mathbf{c}'$ is indeed $d = D$. The base case $m = 1$ follows since the code $\mathcal{C}$ is in fact the original MDS-code $\mathcal{S}$. For the induction step, assume correctness for $m-1$ and we shall prove the claim for m. Let $\mathbf{e}_1, \ldots, \mathbf{e}_K$ be the standard basis of $\mathbb{F}^K$, and observe that there exists $i^* \in [K]$ such that $F(\mathbf{e}_{i^*}, \cdot, \ldots, \cdot) \neq F'(\mathbf{e}_{i^*}, \cdots, \ldots, \cdot)$, or otherwise, by Lemma 4.9, it holds that $F = F'$. Let $\mathcal{C}_{m-1}$ be the code obtained from Construction 4.11 when applied with $m-1$, let $H := F(\mathbf{e}_{i^*}, \cdot, \ldots, \cdot)$ and $H' := F'(\mathbf{e}_{i^*}, \cdot, \ldots, \cdot)$ be symmetric $m-1$ forms, and let $\mathbf{u}$ and $\mathbf{u}'$ be the corresponding codewords in $\mathcal{C}_{m-1}$. Then by the induction hypothesis there exists a set $I \subseteq [n]$ of size d such that $\mathbf{u}[i] \neq \mathbf{u}'[i]$ for all $i \in I$. That is, $F(\mathbf{e}_{i^*}, \mathcal{G}_i, \cdot, \ldots, \cdot) \neq F'(\mathbf{e}_{i^*}, \mathcal{G}_i, \cdot, \ldots, \cdot)$ for every $i \in I$. In particular, it holds that $F(\mathcal{G}_i, \cdot, \ldots, \cdot) \neq F'(\mathcal{G}_i, \cdot, \ldots, \cdot)$ for all $i \in I$, so $\mathbf{c}[i] \neq \mathbf{c}'[i]$ for all $i \in I$. We conclude that the distance of $\mathbf{c}$ and $\mathbf{c}'$ is at least d, as required.

Dimension. We proved that for every distinct symmetric m-forms F and F' the corresponding codewords have distance at least 1 Therefore, each symmetric m-form defines a unique codeword, and by Lemma 4.10 the dimension is

$$\log_q(|\mathbb{F}|^{\binom{m+K-1}{m}}) = 1 + \frac{K-1}{m}.$$

Linearity. For every $i \in [n]$ Consider the map φ_i that takes a symmetric m-form F and returns the $(m-1)$-form $F(\mathcal{G}_i, \cdot, \ldots, \cdot)$. It is not hard to verify that φ_i is a linear map, so $\mathcal{C}$ is indeed a linear code.

Conflict checkability. From now on we assume that $m \geq 2$. We continue by proving that $\mathcal{C}$ is a conflict checkable code. Let $(G_{i,j})_{1 \leq i < j \leq n}$ be the conflict functions of $\mathcal{C}$, and observe that for every $1 \leq i < j \leq n$, and every two symbols F_i and F_j that are $(m-1)$-forms, it holds that (1) if $F_i(\mathcal{G}_j, \cdot, \ldots, \cdot) = F_j(\mathcal{G}_i, \cdot, \ldots, \cdot)$ then by Lemma 4.10 there exists a symmetric m-form F that satisfies $F(\mathcal{G}_i, \cdot, \ldots, \cdot) = F_i(\cdot, \ldots, \cdot)$ and $F(\mathcal{G}_j, \cdot, \ldots, \cdot) = F_j(\cdot, \ldots, \cdot)$, and (2) if there exists a symmetric m-form F that satisfies $F(\mathcal{G}_i, \cdot, \ldots, \cdot) = F_i(\cdot, \ldots, \cdot)$ and $F(\mathcal{G}_j, \cdot, \ldots, \cdot) = F_j(\cdot, \ldots, \cdot)$ then $F_i(\mathcal{G}_j, \cdot, \ldots, \cdot) = F(\mathcal{G}_i, \mathcal{G}_j, \cdots, \ldots, \cdot) = F(\mathcal{G}_j, \mathcal{G}_i, \cdots, \ldots, \cdot) = F_j(\mathcal{G}_i, \cdot, \ldots, \cdot)$. Therefore, $G_{i,j}(F_i, F_j) = \mathsf{NEQ}(F_i(\mathcal{G}_j, \cdot, \ldots, \cdot), F_j(\mathcal{G}_i, \cdot, \ldots, \cdot))$.

To see that the code is conflict checkable, consider any symmetric $(m-1)$-forms $F_1, \ldots, F_n$ that satisfy $F_i(\mathcal{G}_j, \cdot, \ldots, \cdot) = F_j(\mathcal{G}_i, \cdot, \ldots, \cdot)$ for all $i \neq j$. Our goal is to prove that there exists a symmetric m-form that satisfies $F(\mathcal{G}_i, \cdot, \ldots, \cdot) = F_i(\cdot, \ldots, \cdot)$ for all $i \in [n]$. Since the n vectors $\mathcal{G}_1, \ldots \mathcal{G}_n$ span $\mathbb{F}^K$, by Lemma 4.9 there exists a symmetric m-form that satisfies $F(\mathcal{G}_i, \cdot, \ldots, \cdot) = F_i(\cdot, \ldots, \cdot)$ for all $i \in [n]$, as required.

Local-to-global consistency. It remains to prove that the code satisfies local-to-global consistency. By a known property of MDS codes, every K rows of $\mathcal{G}$ are linearly independent. Since $n - d + 1 = N - D + 1 = K$, we note that local-to-global consistency follows from Lemma 4.9. This concludes the proof of the theorem. □

4.3.3 Optimal Comparison-Based Codes from Bilinear Forms

Applying Theorem 4.12 and Corollary 4.13 with an $[N, K, D = N - K + 1]_{|\mathbb{F}|}$ MDS-code and $m = 2$, we obtain an $[n, k, d]_q$ comparison-based t-robust conflict decodable code, with code-length $n = N$, dimension $k = (K + 1)/2$, distance $d = D$ and threshold $t = \lfloor (d - 1)/2 \rfloor$, and it holds that $k = \frac{K+1}{2} = \frac{N-D+2}{2} = \frac{n-d+2}{2}$. Therefore, if $2 < D < N$ then by Lemma 1.12 for an odd D the value of k is optimal, and for an even D the value of k is optimal up to an additive factor of $1/2$.

4.3.4 Polynomial-Time Codes from Bilinear Forms for $t = \lfloor (d - 1)/3 \rfloor$

In this section we present codes with polynomial-time conflict-decoder for the special case of $t = \lfloor (d - 1)/3 \rfloor$.

Theorem 4.14. *Let $\mathbb{F}$ be a finite field, and let $\mathcal{G} \in \mathbb{F}^{N \times K}$ be the generator matrix of an MDS code $[N, K, D = N - K + 1]_{|\mathbb{F}|}$. Let $\mathcal{C}$ be the code obtained from Construction 4.11 with $m = 2$. Then $\mathcal{C}$ is an $[n, k, d]_q$ comparison-based t-robust conflict decodable code with a* polynomial-time *conflict-decoder algorithm, for code-length $n = N$, dimension $k = (K + 1)/2$, distance $d = D$, alphabet size $q = |\mathbb{F}|^K$ and threshold $t = \lfloor (d - 1)/3 \rfloor$.*

Proof. Applying Theorem 4.12 and Corollary 4.13 with an $[N, K, D = N - K + 1]_{|\mathbb{F}|}$ MDS code with $N \geq 4K$ and $m = 2$, we obtain an $[n, k, d]_q$ comparison-based t'-robust conflict decodable code $\mathcal{C}$ with code-length $n = N$, dimension $k = (K + 1)/2$, distance $d = D$, alphabet size $q = |\mathbb{F}|^K$ and threshold $t' = \lfloor (d - 1)/2 \rfloor$, but we will restrict ourselves to threshold $t = \lfloor (d - 1)/3 \rfloor$, and think of the code as a t-robust conflict decodable code. In addition, we are promised that the code $\mathcal{C}$ satisfies local-to-global consistency, and for every $1 \leq i < j \leq n$ the conflict function $G_{i,j}$ is given by $G_{i,j}(F_i, F_j) = \mathsf{NEQ}(F_i(\mathcal{G}_j, \cdot, \ldots, \cdot), F_j(\mathcal{G}_i, \cdot, \ldots, \cdot))$.

The conflict-decoder algorithm. Given a graph K, the algorithm finds a 2-approximation vertex cover E' in K using the classic efficient greedy algorithm, that picks an edge in each step, adds its two vertices to the vertex cover, and removes the two vertices from the graph (For full details, see, e.g., [CLRS09, Section 35.1]). If the size of the vertex cover is more than $2t$, then the algorithm returns no explanation. Otherwise, let $Q \subseteq E'$ be the set of all vertices $u \in E'$ such that there are $K + t$ vertices v outside E' so that (u, v) is not an edge in K. The algorithm sets $E := E' \setminus Q$. If the number of vertices in E is more than t then the algorithm returns no explanation (i.e., it returns an empty list). Otherwise, the algorithm returns a single explanation E.

Analysis. We continue by proving that the code is a t-robust conflict decodable code with respect to the conflict-decoder algorithm that we defined. Observe that $N - K + 1 = d \geq 3t + 1$, and that, since the code is an MDS code, every K rows of $\mathcal{G}$ are a basis of $\mathbb{F}^K$. Whenever E' has size at most $2t$ it holds that (1) there are at least $(n-t) - 2t = n - 3t \geq K$ honest servers outside E' that by Lemma 4.9 fully define a symmetric bilinear form F, and (2) every server in E' that is consistent with at least $K + t$ servers outside E' is consistent with at least K honest servers outside E', and therefore, by Lemma 4.9 is consistent with F.

We continue by proving that $\mathcal{C}$ is a t-robust conflict decodable code with respect to our conflict-decoder algorithm. Fix any $\mathbf{x} \in [q]^n$, any $\mathsf{B} \subseteq [n]$ of size at most t, and any graph K that is B-corrupt with respect to $\mathbf{x}$. To see that validity of explanations holds, assume that the conflict-decoder outputs an explanation E on K, and assume that there are $i, j \in \mathsf{H}$ such that (i, j) is an edge in K. Then without loss of generality i is in E'. We split into cases.

- If the j-th server is not in E' then $F_j(\cdot) = F(\mathcal{G}_i, \cdot)$, and therefore it is impossible that the i-th server is consistent with at least $K + t$ servers outside E', or otherwise $F_i(\mathcal{G}_j) = F(\mathcal{G}_i, \mathcal{G}_j) = F(\mathcal{G}_j, \mathcal{G}_i) = F_j(\mathcal{G}_i)$, so the i-th server is consistent with the j-th server, in contradiction. Therefore, the i-th server is in E, as required.

- If the j-th server is in E' then it is impossible that both the i-th server and the j-th server are consistent with $K+t$ servers outside E' or otherwise $F_i(\mathcal{G}_j) = F(\mathcal{G}_i, \mathcal{G}_j) = F(\mathcal{G}_j, \mathcal{G}_i) = F_j(\mathcal{G}_i)$, so the i-th server is consistent with the j-th server, in contradiction. Therefore, either the i-th server is in E or the j-th server is in E (or both), as required.

To see that the guarantees for good inputs hold, observe that if there exists $\mathbf{c} \in \mathcal{C}$ such that $\mathbf{x}[i] = \mathbf{c}[i]$ for all $i \in \mathsf{H}$, then all the honest servers are pairwise-consistent, the corrupt servers form a vertex cover of size at most $t = K$ in K, and every edge is incident on at least one corrupt server. Therefore, the size of E' is at most $2t$, and the number of honest servers in E' is at most $|E'|/2$. Hence, the number of honest servers outside E' is at least $(N-t) - |E'|/2 \geq N - 2t \geq K + t$. Therefore, all honest servers in E' are consistent with at least $K+t$ servers outside E', which means that E contains no honest servers, and its size is at most t. In addition, by Lemma 4.9 the honest servers fully define the codeword $\mathbf{c}$, as required.

To see that guarantees for bad inputs hold, consider any explanation E (if there is no explanation then we're done). Observe that the number of honest servers outside E is at least $(N-t) - |E| \geq N - 3t \geq K$, and that, by validity of explanations, they are all pairwise consistent, so by Lemma 4.9 they fully define a unique bilinear form F, so the unique codeword that is consistent with the honest servers outside E is the codeword corresponding to F. This concludes the analysis. $\square$

4.3.5 Quasipolynomial-Time Codes from Trilinear Forms

We continue by presenting a code with *quasipolynomial-time* conflict-decoder for a *general* threshold t.

Theorem 4.15. *Let $\mathbb{F}$ be a finite field, let $\mathcal{G} \in \mathbb{F}^{N \times K}$ be the generator matrix of an MDS code $[N, K, D = N - K + 1]_{|\mathbb{F}|}$. Let $\mathcal{C}$ be the code obtained from Construction 4.11 with $m = 3$. Then for every $t \leq \lfloor (d-1)/2 \rfloor$ the code $\mathcal{C}$ is an $[n, k, d]_q$ comparison-based t-robust conflict decodable code for code-length $n = N$, dimension $k = (K+2)/3$, distance $d = D$, alphabet size $q = |\mathbb{F}|^{\binom{K+1}{2}}$ and a conflict-decoder algorithm that runs in time $t^{O(\log t)} \cdot \mathrm{poly}(n)$.*

Proof. Applying Theorem 4.12 and Corollary 4.13 with an $[N, K, D = N - K + 1]_{|\mathbb{F}|}$ MDS code and $m = 3$, we obtain an $[n, k, d]_q$ comparison-based t-robust conflict decodable code $\mathcal{C}$ with code-length $n = N$, dimension $k = (K + 2)/3$, distance $d = D$ and threshold $t = \lfloor (d-1)/2 \rfloor$. In addition, we are promised that the code $\mathcal{C}$ satisfies local-to-global consistency, and for every $1 \leq i < j \leq n$ the conflict function $G_{i,j}$ is given by $G_{i,j}(F_i, F_j) = \mathsf{NEQ}(F_i(\mathcal{G}_j, \cdot, \dots, \cdot), F_j(\mathcal{G}_i, \cdot, \dots, \cdot))$. We continue by proving that $\mathcal{C}$ is a robust conflict decodable code with quasipolynomial-time conflict-decoder algorithm.

The conflict-decoder algorithm. Given a graph K the conflict-decoder does as follows. First, it computes a graph K' by repeatedly removing from K any edge (i, j) so that $|N(i) \cup N(j)| < (N - t) - (K - 1)$, until the property $|N(i) \cup N(j)| \geq (N - t) - (K - 1)$ holds for every edge (i, j). Then it executes Algorithm ExactVC (from Figure 2) on the graph K' and the integer t, and obtains the vertex-covers $(E_1, \dots, E_m)$. Finally, the algorithm outputs the list of explanations $(E_1, \dots, E_m)$.

Analysis. Fix any $\mathbf{x} \in [q]^n$, any $\mathsf{B} \subseteq [n]$ of size at most t, and any graph K that is B-corrupt with respect to $\mathbf{x}$. We first prove the following lemma.

Lemma 4.16. *Let $i, j \in \mathsf{H}$ and let $I \subseteq \mathsf{H} \setminus \{i, j\}$ be a set of at least K honest servers that does not include the i-th and j-th servers, such that the i-th and j-th servers are pairwise consistent with all servers in I. (However, the servers in I are not necessarily pairwise consistent.) Then the i-th server is consistent with the j-th server.*

Proof. For every $\ell \in I$ it holds that $F_i(\mathcal{G}_j, \mathcal{G}_\ell) = F_i(\mathcal{G}_\ell, \mathcal{G}_j) = F_\ell(\mathcal{G}_i, \mathcal{G}_j) = F_\ell(\mathcal{G}_j, \mathcal{G}_i) = F_j(\mathcal{G}_\ell, \mathcal{G}_i) = F_j(\mathcal{G}_i, \mathcal{G}_\ell)$. Since $\mathcal{G}$ is the generator matrix of an MDS code, every K rows of $\mathcal{G}$ span $\mathbb{F}^K$, and therefore, by Lemma 4.9 it holds that $F_i(\mathcal{G}_j, \cdot) = F_j(\mathcal{G}_i, \cdot)$, as required. □

Observe that none of the edges that the conflict-decoder algorithm removes is an edge between two honest servers. To see this it is enough to prove that for any pair of conflicting honest servers $i, j \in \mathsf{H}$ it holds that $|(N(i) \cup N(j)) \cap \mathsf{H}| \geq (N - t) - (K - 1) = N - t - K + 1$ in the original graph K. Assume towards contradiction that $|(N(i) \cup N(j)) \cap \mathsf{H}| \leq N - t - K$. Then there are at least $|\mathsf{H}| - |(N(i) \cup N(j)) \cap \mathsf{H}| = (N - t) - (N - t - K) = K$ honest servers that are pairwise consistent with i and j, and therefore by Lemma 4.16 servers i and j must be consistent, in contradiction. The proof now follows in the same way as the proof of Lemma 1.11, by noting that the modified graph K' is a t-edge-neighborhood graph, as $N - t - K + 1 = (D - 1) - t + 1 \geq 2t - t + 1 = t + 1$, and that by Theorem 3.2 Algorithm ExactVC runs in time $t^{O(\log t)} \cdot \mathrm{poly}(n)$ and the list $(E_1, \dots, E_m)$ contains all t-vertex covers of K'. This concludes the proof of the theorem. □

4.3.6 The Relation to Secret Sharing

A K-out-of-N secret sharing scheme allows a dealer D that holds a secret s to share the secret among N parties, so that every set of $K - 1$ parties has no information about s, but K parties can recover the secret s. We continue with a formal definition of K-out-of-N secret sharing, and explain how our codes can be used to construct *pairwise-verifiable* secret sharing [PC12, Section 3.2.3] (see also [CDM00]).

Definition 4.17. *Let S be a domain of secrets, let R be a domain of randomness, let T be the domain of shares, let N be the number of parties, and let $1 \leq K \leq N$ be a threshold. A K*-out-of-*N* secret sharing scheme *is defined by a sharing algorithm* $\mathsf{share} : S \times R \to T^N$ *and recovery functions* $\mathsf{rec}_A : T^K \to S$ *for all subsets $A \subseteq [N]$ of size at least K, and it satisfies the following properties.*

- (Correctness) *For every $s \in S$, $r \in R$ and a subset $A \subseteq [N]$ of size at least K, it holds that* $\mathsf{rec}_A((s_i)_{i \in A}) = s$, *where* $\mathsf{share}(s, r) = (s_1, \ldots, s_n)$.
- (Perfect privacy) *For every pair of secrets $s, s' \in S$ and for every subset $A \subseteq [n]$ of size at most $K - 1$, it holds that $(s_i)_{i \in A}$ has the same distribution as $(s'_i)_{i \in A}$, where* $\mathsf{share}(s, r) = (s_1, \ldots, s_n)$ *and* $\mathsf{share}(s', r') = (s'_1, \ldots, s'_n)$, *and r, r' are uniformly distributed over R.*

The secret sharing scheme is pairwise verifiable, *if there exists functions $(G_{i,j}, f_{i,j}, f_{j,i})_{1 \leq i < j \leq n}$ such that $G_{i,j}(s_i, s_j) = \mathsf{NEQ}(f_{i,j}(s_i), f_{j,i}(s_j))$, that satisfy the following property. For every subset $A \subseteq [n]$ of size at least K, and for every shares $(s_i)_{i \in A}$, if $G_{i,j}(s_i, s_j) = 0$ for all $i < j \in A$ then there exists $s \in S$ and $r \in R$ such that s_i is the i-th output of* $\mathsf{share}(s, r)$ *for all $i \in A$.*

We continue by presenting a K-out-of-N secret sharing scheme that is pairwise verifiable. Let $\mathcal{G} \in \mathbb{F}^{(N+1) \times K}$ be a generator matrix for an $[N+1, K, D = N - K + 2]_{|\mathbb{F}|}$, where we denote the i-th row of $\mathcal{G}$ by $\mathcal{G}_i$ for $i \in \{0, \ldots, N\}$. Let $\mathcal{C}$ be the $[N+1, k, D]_q$ code defined by Construction 4.11 when applied with $m \geq 2$, where $k = 1 + \frac{K-1}{m}$, and recall that $\mathcal{C}$ satisfies local-to-global consistency (see Theorem 4.12).

Sharing a secret. To share a secret $s \in \mathbb{F}$, sample a random symmetric m-form conditioned on $F(\mathcal{G}_0, \mathcal{G}_0, \ldots, \mathcal{G}_0) = s$. For $i \in [N]$, set the i-th share to be $s_i := F(\mathcal{G}_i, \cdot, \ldots, \cdot)$.

Correctness and pairwise-verifiability. By Lemma 4.9 every K parties fully define a the symmetric m-variate form F, so the parties can recover the m-form F and recover the secret $s = F(\mathcal{G}_0, \mathcal{G}_0, \ldots, \mathcal{G}_0)$. In addition, pairwise-verifiability follows since the code $\mathcal{C}$ satisfies local-to-global consistency and $K = (N+1) - D + 1$.

Privacy. We continue by proving that our scheme is a K-out-of-N secret sharing scheme. We begin with the following claim.

Claim 4.18. *Let $m, t \geq 1$ be integers, let V be a vector space of dimension t, let $\mathbf{v}_1, \ldots, \mathbf{v}_t$ be a basis of V, let $F_1, \ldots, F_{t-1}$ be symmetric $(m-1)$-forms that satisfy $F_i(\mathbf{v}_j, \cdot, \ldots, \cdot) = F_j(\mathbf{v}_i, \cdot, \ldots, \cdot)$ for all $i \neq j \in [t-1]$, and let $s \in \mathbb{F}$. The number of symmetric m-forms F that satisfy $F(\mathbf{v}_i, \cdot, \ldots, \cdot) = F_i(\cdot, \ldots, \cdot)$ for all $i \in [t-1]$ and $F(\mathbf{v}_t, \mathbf{v}_t, \ldots, \mathbf{v}_t) = s$ is exactly 1.*

Proof. We prove the claim by induction on m. The base case $m = 1$ is straightforward. For the induction step, assume correctness for $m-1$, and we shall prove correctness for m. For every symmetric m-form F that satisfies $F(\mathbf{v}_i, \cdot, \ldots, \cdot) = F_i(\cdot, \ldots, \cdot)$ for all $i \in [t-1]$ and $F(\mathbf{v}_t, \mathbf{v}_t, \ldots, \mathbf{v}_t) = s$ it holds that $F_t(\cdot, \ldots, \cdot) := F(\mathbf{v}_t, \cdot, \ldots, \cdot)$ satisfies $F_t(\mathbf{v}_i, \cdot, \ldots, \cdot) = F_i(\mathbf{v}_t, \cdot, \ldots, \cdot)$ for all $i \in [t-1]$ and also satisfies $F_t(\mathbf{v}_t, \mathbf{v}_t, \ldots, \mathbf{v}_t) = s$. By the inductive hypothesis, the number of such F_t is exactly 1. Finally, by Lemma 4.9, for every such F_t there exists a unique symmetric m-form that satisfies $F(\mathbf{v}_i, \cdot, \ldots, \cdot) = F_i(\cdot, \ldots, \cdot)$ for all $i \in [t]$. This concludes the proof of the claim. $\square$

Consider any set I of $K-1$ parties and any two secrets $s, s' \in \mathbb{F}$. To prove that privacy holds it is enough to note that the following claims hold.

- For any symmetric $(m-1)$-forms $(F_i)_{i\in I}$ that satisfy $F_i(\mathcal{G}_j, \cdot, \ldots, \cdot) = F_j(\mathcal{G}_i, \cdot, \ldots, \cdot)$ for all $i, j \in I$, the number of symmetric m-forms F that satisfy $F(\mathcal{G}_i, \cdot, \ldots, \cdot) = F_i(\cdot, \ldots, \cdot)$ and $F(\mathcal{G}_0, \mathcal{G}_0, \ldots, \mathcal{G}_0) = s$ is equal to the number of symmetric m-forms F that satisfy $F(\mathcal{G}_i, \cdot, \ldots, \cdot) = F_i(\cdot, \ldots, \cdot)$ and $F(\mathcal{G}_0, \mathcal{G}_0, \ldots, \mathcal{G}_0) = s'$. This now follows immediately from Claim 4.18.
- The number of m-forms F that satisfy $F(\mathcal{G}_0, \ldots, \mathcal{G}_0) = s$ is equal to the number of m-forms F that satisfy $F(\mathcal{G}_0, \ldots, \mathcal{G}_0) = s'$. Indeed, let M be the number of m-forms (this number was computed in Lemma 4.10, but we don't need the exact value). Observe that (1) by Lemma 4.9 every m-form F is fully determined by $(F(\mathcal{G}_i, \cdot \ldots, \cdot))_{i\in I\cup\{0\}}$, (2) by Claim 4.18, given $(F(\mathcal{G}_i))_{i\in I}$ the number of m-forms F that satisfy $F(\mathcal{G}_0, \ldots, \mathcal{G}_0) = s$ is exactly 1, and therefore, the number of m-forms F that satisfy $F(\mathcal{G}_0, \ldots, \mathcal{G}_0) = s$ is exactly $M/|\mathbb{F}|$. Since the same argument works for s' as well, the claim follows.

This concludes the proof of privacy.

5 Round-Optimal Statistical MPC with Strong Honest Majority

In this section we consider the scenario where n parties $\mathsf{P} = \{P_1, \ldots, P_n\}$ wish to compute a function of their joint inputs with *information-theoretic* security, at the presence of a computationally-unbounded active (aka Byzantine or malicious) rushing adversary that controls up to t of the parties. We assume that each pair of parties is connected by a secure and authenticated point-to-point channel, and that all parties have access to a common broadcast channel.[8] Throughout this section, we let κ denote a statistical security parameter, and assume without loss of generality that $\kappa = \omega(\log n)$. We also let $\mathbb{F}$ be a finite field, and assume that it is sufficiently large, $|\mathbb{F}| \geq 2^{\Omega(\kappa)}$. We denote by B the set of corrupt parties, by H the set of honest parties, and we let $1, \ldots, n$ be n distinct elements field elements.

We present a *quasipolynomial-time* 3-round MPC protocol with statistical security with $n \geq 3t+1$, and a *polynomial-time* 3-round MPC protocol with statistical security with $n \geq 3(1+\epsilon)t$ for any constant $\epsilon > 0$. We first design a 2-round verifiable secret sharing protocol (VSS), and then simply plug our VSS protocol in the satatistical framework of [AKP20] to obtain a 3-round protocol for general MPC.

5.1 Verifiable Secret Sharing

In this section we present a 2-round protocol for verifiable secret sharing. Our construction follows the blueprints of the (exponential-time) construction of [AKP20], that implicitly used the (exponential-time) t-robust conflict decodable codes based on *symmetric bilinear forms* (more concretely, they used symmetric bivaraite polynomials). We replace the symmetric bilinear forms with the quasipoylnomial-time codes from *symmetric trilinear forms* to improve the efficiency of the construction. To make this section self-contained, and in accordance with the literature of secure multiparty computation, it would be easier to consider a concrete instance of the codes that

[8] For a formal definition of secure multiparty computation in our settings, see, e.g., [AKP20, Appendix A].

is based on *trivariate polynomials*. We emphasize that the protocol can be instantiated with any t-robust conflict decodable codes obtained from Theorem 4.15.

The rest of the section is organised as follows. In Section 5.1.1 we briefly discuss trivariate polynomials. In Section 5.1.2 we present the notion of *interactive signatures*, that will be used as a basic building block in our construction. In Section 5.1.3 we use interactive signatures to construct *weak-commitments*, which is a weaker notion of VSS. Finally, in Section 5.1.4 we use weak commitments to construct a full-fledged VSS protocol. Throughout this section, we always consider the optimal resiliency $n \geq 3t+1$ and obtain a quasipolynomial-time protocol. In Section 5.1.5, we explain how to modify the VSS protocol when $n \geq (3+\epsilon)t$ in order to obtain a *polynomial-time* protocol.

5.1.1 On Trivariate polynomials

Let $\mathcal{C}$ be the code obtained from Construction 4.11 when it is instantiated with $[n, t+1, d = n-t]_{|\mathbb{F}|}$ Reed-Solomon code and $m = 3$. One can verify that

$$\mathcal{C} = \left\{ (F(x,y,1), \ldots, F(x,y,n)) \;\middle|\; \begin{array}{c} F \text{ is a symmetric trivariate polynomial} \\ \text{of degree at most } t \text{ in each variable} \end{array} \right\}$$

Then, by Theorem 4.15, for any $n \geq 3t+1$ the code $\mathcal{C}$ is a t-robust conflict decodable code, with conflict-decoder algorithm with time $t^{O(\log t)} \cdot \mathrm{poly}(n)$. We recall that the i-th server, that holds the bivariate polynomial $f_i(x,y)$, is consistent with the j-th server, that holds the bivariate polynomial $f_j(x,y)$ if and only if $f_i(x,j) = f_j(x,i)$.

Remark 5.1 (On punctured codes)**.** *Let $I \subseteq [n]$ be a set of size $n-\ell$ for some $0 \leq \ell \leq t$. In some cases it will be useful to consider the punctured code*

$$\mathcal{C}_I = \left\{ (F(x,y,i))_{i \in I} \;\middle|\; \begin{array}{c} F \textit{ is a symmetric trivariate polynomial} \\ \textit{of degree at most } t \textit{ in each variable} \end{array} \right\},$$

that is, $\mathcal{C}_I$ is the code $\mathcal{C}$ punctured at the indices not in I. Again, $\mathcal{C}_I$ can be obtained from Construction 4.11 by instantiating it with $[n-\ell, t+1, n-\ell-t]_{|\mathbb{F}|}$ Reed-Solomon code, and $m = 3$, and therefore, by Theorem 4.15, for any $n \geq 3t+1$ the code $\mathcal{C}_I$ is a $(t-\ell)$-robust conflict decodable code, with conflict-decoder with time $t^{O(\log t)} \cdot \mathrm{poly}(n)$.

In addition, the following fact is analogous to Lemma 4.9.

Fact 5.2. *Let $m \geq 1$ be an integer. Let $K \subseteq \{1, \ldots, n\}$ be a set of size at least $t+1$, and let $(f_k(x_1, \ldots, x_{m-1}))_{k \in K}$ be a set of symmetric bivariate polynomials of degree at most t in each variable. If for every $i, j \in K$ it holds that $f_i(x_1, \ldots, x_{m-2}, j) = f_j(x_1, \ldots, x_{m-2}, i)$ then there exists a unique symmetric m-variate polynomials $F(x_1, \ldots, x_m)$ of degree at most t in each variable such that $f_k(x_1, \ldots, x_{m-1}) = F(x_1, \ldots, x_{m-1}, k)$ for every $k \in K$.*

5.1.2 Interactive Signature

Our first building block is *interactive signatures*, presented in Definition 5.3, taken verbatim from [AKP20].

Definition 5.3 (Interactive Signature Scheme (ISS)). *In an interactive signature scheme (ISS) amongst a set* P *of* n *parties, there are three distinguished parties, a dealer* $D \in \mathsf{P}$*, an intermediary* $I \in \mathsf{P}$*, and a receiver* $R \in \mathsf{P}$*. In addition, all parties in* P *play the role of* verifiers*. At the beginning of the protocol,* D *holds an input* $s \in \mathbb{F}$*, referred to as the* secret*, and each party (including the dealer) holds an independent random string. The protocol consists of two phases, a* distribute *phase, and a* verify & open *phase with the following syntax.*

- Distribute: *In this phase,* D *sends* s *to a designated intermediary* $I \in \mathsf{P}$*.* D *also sends private information (computed based on its secret and randomness) to* I *and to each of the verifiers in* P*.*
- Verify & open: *This phase consists of two parts, verification and opening.*
  - *In the verification, the parties communicate in order to ensure that the information received from* D *are consistent. The verification ends with a public accept or reject, indicating whether the verification is successful or not.*
  - *In the opening,* I *sends* s *to the receiver* R*, and all verifiers send information to* R *in order to make sure that* R *accepts only the correct value* s*.*

  If the verification failed, then R *outputs* $\perp$*. Otherwise, upon a successful verification,* R *verifies that the value* $s' \in \mathbb{F}$ *received from* I *is valid, using the information received from the verifiers in the opening. If* s' *is valid then* R *outputs* s'*, otherwise* R *outputs* $\perp$*.*

A two-phase, n*-party protocol as above is called a* $(1-\epsilon)$*-secure ISS scheme, if for any adversary* $\mathcal{A}$ *corrupting at most* t *parties amongst* P*, the following holds:*

- *Correctness: If* D *and* I *are honest, the verify phase will complete with a success and an honest* R *accepts and outputs* s *in the open phase.*
- ϵ*-nonrepudiation: Assume that* I *and* R *are honest. Then the probability that the verification succeeds and* R *does not accept the value* s' *sent by* I *in the opening is at most* ϵ*.*
- ϵ*-unforgeability: Assume that* D *and* R *are honest and let* View *be any possible view of the adversary in the ISS execution. Then, conditioned on* View*, the probability that* R *outputs either* s *or* $\perp$ *is at least* $1-\epsilon$*.*
- *Privacy: If* D, I *and* R *are honest, then the distribution of the adversary's view is identical for any two secrets* s *and* s'*. Denoting* View_s *as* $\mathcal{A}$*'s view during the ISS scheme when* D*'s secret is* s*, the privacy property demands* $\mathsf{View}_s \equiv \mathsf{View}_{s'}$ *for any* $s \neq s'$*.*
- *Output extraction: In any execution where* D *is corrupt and* R *is honest, the output of* R *can be extracted from the view* View *of the corrupt parties .*

The work of [AKP20] presented a polynomial-time protocol iSig for interactive signatures, where each phases requires one round. Formally, they proved the following lemma. (See [AKP20, Lemma 4.2].)

Lemma 5.4. *Let* κ *be the security parameter, let* n *be the number of parties, and let* t *be the number of corrupt parties, so that* $n \geq 3t+1$*. Protocol* iSig *is* $(1-2^{-\kappa})$*-secure interactive signature scheme, tolerating a static, active rushing adversary corrupting* t *parties. Moreover, the protocol achieves perfect privacy and perfect correctness, and can be implemented in time* $\mathrm{poly}(n, \kappa, \log|\mathbb{F}|)$*.*

5.1.3 Weak Commitment

Following the blueprints of [AKP20], we continue by designing a protocol for *weak commitment*. At a high level, weak commitment allows a dealer $D \in \mathsf{P}$ to share a symmetric trivariate polynomial $H(x, y, z)$ of degree at most t in each variable among the parties in P, so that party P_i receives the symmetric bivariate polynomial $H(x, y, i)$. More precisely, if D is honest, then every party P_i learns *only* its share $H(x, y, i)$. If D is corrupt, then D can cause a small set P' of honest parties to output a special failure symbol $\perp$; however, every other honest P_i outputs $H_i(x, y)$ so that all polynomials $(H_i(x, y))_{P_i \in \mathsf{H} \setminus \mathsf{P}'}$ are consistent with some symmetric trivariate polynomial $H(x, y, z)$ of degree at most t in each variable. This is formalized in the following functionality.

Functionality $\mathcal{F}_{\mathsf{wcom}}$

- **Input:** D inputs a symmetric trivariate polynomial $H(x, y, z)$ of degree at most t in each variable. A corrupt D also inputs a bit flag, and a set P' of size at most $n - 2t - 1$.
- **Outputs:** If D is honest then the functionality returns the bivariate polynomial $H(x, y, i)$ to P_i.

  If D is corrupt and $\mathsf{flag} = 1$ then the functionality returns "D is corrupt" to all the parties. Otherwise, the functionality returns $\perp$ to every $P_i \in \mathsf{P}'$, and returns the bivariate polynomial $H(x, y, i)$ to every $P_i \in \mathsf{P} \setminus \mathsf{P}'$.

Figure 5: Functionality $\mathcal{F}_{\mathsf{wcom}}$

The protocol. Our protocol follows the blueprints of [AKP20]. In the first round the dealer sends the bivariate polynomial $h_i(x, y) := h(x, y, i)$ to P_i where x and y are treated as formal variables. Consider the code $\mathcal{C}$ discussed in Section 5.1.1, and observe that P_i holds the i-th entry of the codeword defined by $h(x, y, z)$, and we therefore think of P_i as the i-th server of the robust conflict decodable code, where the set of corrupt servers is exactly the set of corrupt parties. In addition, for every pair of parties (P_i, P_j) the dealer executes the distribute phase of iSig with the value $h(x, j, i)$ and with P_i as the intermediate and P_j as the receiver.

In the second round, every pair of parties (P_i, P_j) performs a secure public consistency check, and completes the verify and open phase of iSig. Recall that P_i is consistent with P_j if and only if $h_i(x, j) = h_j(x, i)$. Therefore, the consistency check is executed as follows. In the first round P_i and P_j exchange a random pad $r_{ij}(x)$ which is a random degree-t univariate polynomial, and in the second round P_i broadcasts $h_i(x, j) + r_{i,j}(x)$ and P_j broadcasts $h_j(x, i) + r_{i,j}(x)$. Observe that P_i is consistent with P_j if and only if the broadcast messages are equal. For honest P_i and P_j, all parties learn whether P_i is consistent with P_j. However, if, e.g., P_i is corrupt, then P_i can fully control the result of the consistency check, but this is not a problem, as this behaviour is also allowed for a corrupt server. We also note that when D, P_i and P_j are honest this public comparison reveals no information about $h(x, j, i)$, as all the other parties can see is a random degree-t polynomial, so privacy is preserved. We emphasize the the public comparison stage strongly relies on the fact that the code $\mathcal{C}$ is a comparison-based code.

At this stage the parties can locally compute the conflict graph based on the public comparisons, and find an explanation E of size t using the conflict-decoder (if no such explanation exists then the parties deduce that D is corrupt). Every party outside the set E simply outputs the share it received from the dealer, while every party P_i in E recovers its own share by interpolating all the univariate polynomials $h(x, i, j)$ for every P_j that successfully opened its signature. If the results of the interpolation is not a symmetric bivariate polynomial of degree-t, then P_i outputs $\perp$. The protocol is described in Figure 6.

Protocol swcom

Inputs: D holds a symmetric trivariate polynomial $h(x, y, z)$ of degree at most t in each variable. All parties share a statistical security parameter 1^κ.

R1: D and every party P_i do the following in parallel.

1. For every ordered pair (P_i, P_j), D initiates the distribute phase of $(n+1)$ instances of iSig, denoted as $(\mathsf{iSig}_{i,j,k})_{k=0,\ldots,n}$, with P_i as the intermediary, P_j as the receiver and $(h(k, j, i))_{k=0,\ldots,n}$ as the secret (and with security parameter 1^κ).
2. Each P_i picks a symmetric bivariate polynomial $r_i(x, y)$ of degree at most t in each variable, and sends the univariate polynomial $r_{i,j}(x) := r_i(x, j)$ to every P_j.
3. Each P_i sets its tentative share to be $h_i(x, y)$.

R2: The parties do as follows.

1. For each ordered pair (i, j), party P_i broadcasts $m_i(x, y) := h_i(x, y) + r_i(x, y)$, and P_j broadcasts $m_{i,j} := h_j(x, i) + r_{i,j}(x)$.
2. For each ordered pair (i, j), the parties execute the verify and open phases of $(\mathsf{iSig}_{i,j,k})_{k=0,\ldots,n}$ and let $h'_{i,j,k}$ be the output of $\mathsf{iSig}_{i,j,k}$. If $(h'_{i,j,k})_{k=0,\ldots,n}$ correspond to a univariate degree-t polynomial then P_j sets $h'_{i,j}(x)$ to be the corresponding polynomial, and otherwise P_j sets $h'_{i,j} := \perp$.

Local Computation: Each party does as follows.

1. A pair (P_i, P_j) is called *conflicting* pair if $m_i(x, j) \neq m_{i,j}(x)$ or $m_j(x, i) \neq m_{j,i}(x)$. We say that P_i is conflicted with D if (D, P_i) is a conflicting pair, or if any of the $\mathsf{iSig}_{i,j,k}$ instances for $j \in [n]$ and $k \in \{0, \ldots, n\}$ results in `Failure`. Construct the conflict graph, and remove from it any party in conflict with D (let ℓ be the number of removed parties) to obtain a graph K. Let L be the set of parties removed from the graph, and let $M := \mathsf{P} \setminus L$.
2. Execute the conflict-decoder algorithm of the punctured code $\mathcal{C}_M$ on K to obtain a list of explanations $(E_1, \ldots, E_m)$, where each explanation of size at most $t - \ell$ (see Remark 5.1).
3. If no explanation exists (and in particular, if $\ell > t$) then the parties output "D is corrupt" and terminate. Otherwise, let E be any one of those explanations.[a]
4. Let $\mathsf{W} := \mathsf{P} \setminus (E \cup L)$. Every $P_i \in \mathsf{W}$ outputs $h_i(x, y)$. Every $P_i \notin \mathsf{W}$ computes a polynomial $h'_i(x, y)$ interpolating over $(h'_{j,i}(x))_{j:P_j \in \mathsf{W}, h'_{j,i} \neq \perp}$. If $h'_i(x, y)$ is not a symmetric bivariate polynomial of degree t in each variable then P_i outputs $\perp$. Otherwise, P_i outputs $h'_i(x, y)$.

[a] Looking forward, when using wcom as a subprotocol, this explanation will be chosen by the outer protocol.

Figure 6: Protocol swcom

The following Theorem shows that protocol wcom is a UC-secure implementation of $\mathcal{F}_{\mathsf{wcom}}$.

Theorem 5.5. *Let κ be the security parameter, let n be the number of parties, and let t be the number of corrupt parties, so that $n \geq 3t+1$. Protocol* wcom *is a UC-secure implementation of $\mathcal{F}_{\mathsf{wcom}}$ with statistical security against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is* $\mathrm{poly}(t^{\log t}, n, \kappa, \log |\mathbb{F}|)$.

Since the security analysis is the same as in [AKP20], we omit the full proof, and only sketch the proof of correctness here.

Proof sketch. We split into cases.

Honest D. We need to prove that in a real-world execution with input $h(x, y, z)$ to the honest dealer, every honest party P_i outputs $h(x, y, i)$. Observe that only corrupt parties are conflicted with D, and by the guarantees for good inputs of robust conflict decodable codes, we are promised that an explanation exists, so D is not discarded. Every honest party inside W (which is not necessarily the set of all honest parties) outputs $h(x, y, i)$. In addition, for every honest P_i not in W, the correctness of iSig guarantees that $h'_{j,i}(x) = h(x, i, j)$ for every honest P_j, and that with all but negligible probability, $h_{j,i}$ is either $\perp$ or equals to $h(x, i, j)$ for every corrupt P_j. Since there are at least $(n - t) - |E| \geq t + 1$ honest parties in W, and by Fact 5.2, we conclude that $h'_i(x, y) = h(x, y, i)$, as required. Therefore every honest party P_i outputs $h(x, y, i)$, as required.

Corrupt D. If D is discarded then correctness holds and the parties output "D is corrupt". Otherwise, by the validity of explanations of the robust conflict decodable codes, for every explanation E and every pair of conflicting honest parties P_i and P_j, either $P_i \in E$ or $P_j \in E$. In addition, the set W contains at least $(n - t) - \ell - |E| \geq t + 1$ pairwise consistent honest parties, that by Fact 5.2 define a unique symmetric trivariate polynomial $h(x, y, z)$ of degree at most t in each variable[9], and every honest P_i in W outputs $h(x, y, i)$. For every honest $P_i \notin W$, by the nonrepudiation property of iSig, with all but negligible probability it holds that $h_{j,i}(x) = h(x, i, j)$ for every honest $P_j \in \mathsf{W}$. Since there are at least $t + 1$ honest parties in W, the polynomial that P_i recovers is either $h(x, y, i)$ or $\perp$. Finally, observe that the number of honest parties with non-$\perp$ output is at least $|\mathsf{W}| \geq t + 1 = (n - t) - (n - 2t - 1)$, as required. This completes the proof sketch. $\square$

Remark 5.6 (On tentative shares)**.** *We note that for every honest party in* W*, the tentative share from the end of Round 1 becomes its final share at the end of Round 2. For honest parties outside* W *with non-*$\perp$ *output, the tentative share might change only if the dealer is corrupt.*

5.1.4 The VSS Protocol

In this section we present the verifiable secret sharing protocol. At a high level, the VSS functionality takes a symmetric trivariate polynomial $F(x, y, z)$ from the dealer D, and returns the bivariate polynomial $F(x, y, i)$ to P_i. We continue with a formal definition of the functionality.

Functionality $\mathcal{F}_{\mathsf{vss}}$

Inputs.

- An honest D inputs a symmetric trivariate polynomial $F(x, y, z)$ of degree t in each variable.
- A corrupt D inputs a polynomial $F(x, y, z)$.

[9] More generally, this follows from the guarantees for bad inputs of robust conflict decodable codes.

Outputs.

- For an honest D, the functionality returns the bivariate polynomial $f_i(x, y) := F(x, y, i)$ to every party P_i.
- For a corrupt D, if the input $F(x, y, z)$ is not a symmetric trivariate polynomial $F(x, y, z)$ of degree t in each variable, the functionality resets $F(x, y, z)$ to be the zero-polynomial. The functionality $\mathcal{F}_{\mathsf{vss}}$ returns the univariate polynomial $f_i(x, y) := F(x, y, i)$ to every P_i.

Figure 7: Functionality $\mathcal{F}_{\mathsf{vss}}$

The protocol. Our VSS protocol follows the blueprint of [AKP20]. At a high level we follow the same path of Protocol wcom, except that now every random pad will also be committed via wcom. This will allow every party that is not in W to recover its share even if D is corrupt. The protocol is presented in Figure 8.

Protocol vss

Inputs: D holds a symmetric trivariate polynomial $F(x, y, z)$ of degree at most t in each variable. All parties share a statistical security parameter 1^κ.

R1 D and every party P_i do the following in parallel.

1. D sends to each P_i the bivariate polynomial $f_i(x, y) := F(x, y, i)$.
2. Each party P_i picks a random symmetric trivariate polynomial $h_i(x, y, z)$ of degree at most t in each variable and initiates an instance of swcom, denoted as swcom_i as a dealer with polynomial $h_i(x, y, z)$.

R2 For each ordered pair (i, j), P_i broadcasts the bivariate polynomial $p_i(x, y) := f_i(x, y) + h_i(x, y, 0)$ and P_j broadcasts the univariate polynomial $p_{i,j}(x) := f_j(x, i) + h_{i,j}(x, 0)$, where $h_{i,j}(x, y)$ is the tentative share of P_j in swcom_i. In parallel, parties execute **R2** of swcom_i for all $i \in \{1, \ldots, n\}$.

Local Computation All parties do as follows.

1. A pair (P_i, P_j) is called *VSS-conflicting* pair if $p_i(x, j) \neq p_{i,j}(x)$ or $p_j(x, i) \neq p_{j,i}(x)$. Construct the conflict graph, and remove from it any party in conflict with D (let ℓ be the number of removed parties) to obtain a graph K. Let L be the set of parties that were removed from the graph.

   Let $\mathsf{K}_1, \ldots, \mathsf{K}_n$ be the graphs computed in $\mathsf{swcom}_1, \ldots, \mathsf{swcom}_n$, respectively, and let $L_1, \ldots, L_n$ be the corresponding sets of parties that were removed from the graphs.
2. Every party executes Algorithm Find$E$ from Figure 9 on inputs $(\mathsf{K}, \mathsf{K}_1, \ldots, \mathsf{K}_n, L, L_1, \ldots, L_n)$, to obtain the sets $E, E_1, \ldots, E_n, I$ that satisfy the following requirements:
   (a) E is an explanation of K. Let $\mathsf{W} := \mathsf{P} \setminus (E \cup L)$.
   (b) $I \subseteq \mathsf{W}$ is a set of size at least $2t + 1$,
   (c) for every $i \in I$, the set E_i is an explanation of K_i. Let $\mathsf{W}_i := \mathsf{P} \setminus (E_i \cup L_i)$.
   (d) for every $i \in I$ it holds that $|\mathsf{W} \cap \mathsf{W}_i| \geq 2t + 1$.
3. If Algorithm Find$E$ returns `Failure` then D is discarded, each party resets its share to be the all-zero polynomial, and terminates.

4. Otherwise, for every $P_i \in \mathsf{W}$ the parties locally complete the local computation of swcom_i with the set E_i as the explanation.
5. Every $P_i \in \mathsf{W}$ outputs $f_i(x, y)$ and terminates.
6. Every $P_i \notin \mathsf{W}$ computes the set W'_i of indices j such that P_i has non-$\perp$ output in swcom_j, and resets the bivaraite polynomial $f_i(x, y)$ to the polynomial interpolated over the univariate polynomials $(p_j(x, i) - h_{j,i}(x, 0))_{P_j \in I \cap \mathsf{W}'_i}$ (where $p_j(x, y)$ was broadcasted by P_j in **R2** and $h_{j,i}(x, y)$ is the final share of P_i in swcom_j). Finally, P_i outputs $f_i(x, y)$.

Figure 8: Protocol vss

Finding $E, E_1, \ldots, E_n, I$. We continue with the description of Algorithm Find$E$.

Algorithm Find$E$

Input: The conflict graphs $\mathsf{K}, \mathsf{K}_1, \ldots, \mathsf{K}_n$, amd the corresponding sets $L, L_1, \ldots, L_n$ of parties that were removed from those graphs.

The algorithm:

1. Let $M := \mathsf{P} \setminus L$. Execute the conflict-decoder algorithm of the punctured code $\mathcal{C}_M$ on K to obtain the explanations $S_1, \ldots, S_m$ of size at most $t - |L|$ (See Remark 5.1).
2. For $i = 1, \ldots, m$:
   (a) Initialize $I_i := \varnothing$.
   (b) For $j \in [n] \setminus (S_i \cup L)$ do as follows:
      i. Let $M_j := \mathsf{P} \setminus L_j$. Execute the conflict-decoder algorithm of the punctured code $\mathcal{C}_{M_j}$ on K_j to obtain the explanations $S_{j,1}, \ldots, S_{j,m_j}$ of size at most $t - |L_j|$ (See Remark 5.1).
      ii. If there exists some index k such that $n - |S_i \cup L \cup S_{j,k} \cup L_j| \geq 2t + 1$ then set $S^*_{i,j} := S_{j,k}$, and add the index j to the set I_i.
   (c) If $|I_i| \geq 2t + 1$ then:
      i. Set $E := S_i$.
      ii. Set $E_j := S^*_{i,j}$ for $j \in I_i$, and $E_j = \varnothing$ otherwise.
      iii. Set $I := I_i$.
      iv. Return $(E, E_1, \ldots, E_n, I)$.
      (Otherwise, continue to the next iteration.)
3. Return `Failure`.

Figure 9: Algorithm Find$E$

Observe that the running time of the algorithm is polynomial in the running time of the conflict-decoder algorithm, and therefore it is $t^{O(\log t)} \cdot \mathrm{poly}(n)$. It is straightforward to verify that if the Algorithm does not return `Failure`, then the algorithm returns sets $(E, E_1, \ldots, E_n, I)$ that satisfy the requirements mentioned in Protocol vss. We continue by proving that if D is honest then the algorithm does not return `Failure`.

Claim 5.7. *Assume that D is an honest dealer. Then Algorithm FindE on $(\mathsf{K}, \mathsf{K}_1, \ldots, \mathsf{K}_n, L, L_1, \ldots, L_n)$ finds sets $E, E_1, \ldots, E_n, I$ that satisfy the requirements with probability 1.*

Proof. Since D is honest, no honest party is conflicted with D, so L contains no honest party. Therefore, by the guarantees for good inputs of the robust codes, we are promised that the conflict-decoder algorithm on K returns an explanation S_i such that $S_i \subseteq \mathsf{B}$. Therefore, for every honest party P_j it holds that $j \in [n] \setminus (S_i \cup L)$, and since there are $n - t \geq 2t + 1$ honest parties, it is enough to prove that for every honest P_j the algorithm finds an explanation $S^*_{i,j}$ in K_j, so $j \in I_i$.

For every honest P_j, the parties that are removed from K_j in the execution of wcom_j are either (1) parties that are conflicted with P_j as a dealer in wcom_j, and those parties are necessarily corrupt, or (2) parties P_k for whom the verification of the iSig execution with P_j as the dealer and P_k as the intermediate ended with `Failure`, and again, by the (perfect) correctness of iSig, those parties are necessarily corrupt. Therefore, the set L_j contains no honest parties. By the guarantees for good inputs, we are promised that the conflict-decoder algorithm on K_j returns an explanation $S_{j,k}$ such that $S_{j,k} \subseteq \mathsf{B}$. Since $S_i, L, S_{j,k}, L_j \subseteq \mathsf{B}$, we conclude that $n - |S_i \cup L \cup S_{j,k} \cup L_j| \geq 2t + 1$, and therefore the algorithm adds the index j to I_i. This concludes the proof of the claim. □

Security of vss. We continue by arguing that vss is a a UC-secure implementation of $\mathcal{F}_{\mathsf{vss}}$.

Theorem 5.8. *Let κ be the security parameter, let n be the number of parties, and let t be the number of corrupt parties, so that $n \geq 3t + 1$. Protocol vss is a UC-secure implementation of $\mathcal{F}_{\mathsf{vss}}$ with statistical security against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is* $\mathrm{poly}(t^{\log t}, n, \kappa, \log |\mathbb{F}|)$.

Since the security analysis is the same as in [AKP20], we omit the full proof, and only sketch the proof of correctness here.

Proof sketch. We split into cases.

Honest D. As argued in Claim 5.7, when D is honest the Algorithm Find$E$ always finds sets $(E, E_1, \ldots, E_n, I)$ that satisfy the requirements, so D is not discarded. Let $(E, E_1, \ldots, E_n, I)$ be the sets that the algorithm outputs, and let $\mathsf{W} := \mathsf{P} \setminus (E \cup L)$ and $\mathsf{W}_i := \mathsf{P} \setminus (E_i \cup L_i)$. Observe that every honest P_i in W outputs $f_i(x, y) = F(x, y, i)$, as required. Consider now any honest $P_i \notin \mathsf{W}$, and let P_j be a party in $I \cap \mathsf{W}'_i$.

- Assume that P_j is honest. Then, by the correctness of wcom_j, it holds that $p_j(x, i) - h_{j,i}(x, 0) = f_j(x, i) + h_j(x, i, 0) - h_j(x, i, 0) = F(x, i, j)$ (with all but negligible probability).
- Assume that P_j is corrupt. The set W_j contains at least $2t+1$ parties, and therefore it contains at least $2t + 1 - t = t + 1$ honest parties. By the correctness of wcom_j, with all but negligible probability, the shares of those honest parties fully define a symmetric trivariate polynomial $h_j(x, y, z)$ of degree at most t in each variable.

  Consider any honest P_k in $\mathsf{W} \cap \mathsf{W}_j$, and recall that the tentative share of P_k in wcom_j becomes the final share of P_k (see Remark 5.6). Since P_j and P_k are in W they are consistent, and therefore it must hold that $p_j(x, k) = F(x, k, j) + h_j(x, k, 0)$. That is, P_j has broadcasted a polynomial $p_j(x, y)$ so that $p_j(x, k)$ is equal to $F(x, k, j) + h_j(x, k, 0)$ for any honest P_k in $\mathsf{W} \cap \mathsf{W}_j$. Since $\mathsf{W} \cap \mathsf{W}_j$ is of size at least $2t + 1$ it must contain at least $2t + 1 = t + 1$ honest parties, so necessarily $p_j(x, y) = F(x, y, j) + h_j(x, y, 0)$. By the correctness of swcom_j, with all but negligible probability P_i holds $h_{j,i}(x, y) = h_j(x, y, i)$, and we conclude that P_i recovers the share $p_j(x, i) - h_{j,i}(x, 0) = F(x, i, j)$.

Finally, since I is of size at least $2t+1$ it must contain at least $2t+1-t = t+1$ honest parties. Since W'_i contains all honest parties, we conclude that P_i recovers at least $t+1$ shares, and so P_i correctly recovers $F(x,y,i)$.

Corrupt D. If D is discarded then we are done. Otherwise, let $\mathsf{W_H} := \mathsf{W} \cap \mathsf{H}$, observe that $|\mathsf{W_H}| \geq |\mathsf{W}| - |\mathsf{B}| \geq 2t+1-t = t+1$, and that all parties in $\mathsf{W_H}$ are pairwise consistent, so the shares of the parties in $\mathsf{W_H}$ define a symmetric trivariate polynomial $F(x,y,z)$ of degree at most t in each variable. It is enough to show that the shares of all honest parties outside W is also consistent with $F(x,y,z)$. The proof now follows by following the same lines as in the case of honest D. This concludes the proof. □

5.1.5 Polynomial-Time VSS for $n \geq (3+\epsilon)t$

Let $\epsilon > 0$ be any constant. We note that if $n \geq (3+\epsilon)t$ then we can obtain a polynomial time VSS protocol. The main observation is that the bottleneck of the running-time of both wcom and vss is the conflict-decoder algorithm that runs in time $t^{O(t)} \cdot \mathrm{poly}(n)$. We note that when $n \geq (3+\epsilon)t$ it suffices to have weaker guarantees from the conflict-decoder algorithm to obtain a VSS protocol. Those weaker guarantees will allows us to obtain a polynomial time weak conflict decoder. We continue by describing the requirements from the weak conflict-decoder algorithm.

Relaxing the requirements. Let $\delta = \epsilon/100$. The weak conflict-decoder should satisfy the same guarantees as in Definition 1.9 with the following relaxations.

- The size of each explanation can be at most $(1+\delta)t$.
- The guarantees for good inputs are relaxed as follows: If there exists a codeword $\mathbf{c} \in \mathcal{C}$ such that $\mathbf{x}[i] = \mathbf{c}[i]$ for every $i \in \mathsf{H}$, then (1) there exists an explanation E such that $|E \setminus \mathsf{B}| \leq 2\delta t$ (in particular $\mathcal{L}$ is not empty), and (2) for every explanation E, the codeword $\mathbf{c}$ is the only codeword that satisfies $\mathbf{x}[i] = \mathbf{c}[i]$ for all $i \in \mathsf{H} \setminus E$.

Validity of explanations and guarantees for bad inputs remain the same.

The weak conflict-decoder. We continue with a description of a weak conflict-decoder that satisfies the relaxed requirements with respect to the code $\mathcal{C}$ that is based on trivariate polynomials. Given a graph K the conflict-decoder does as follows. First, it computes a graph K' by repeatedly removing from K any edge (i,j) so that $|N(i) \cup N(j)| < (n-t)-t$, until the property $|N(i) \cup N(j)| \geq n-2t$ holds for every edge (i,j). Then it executes the Algorithm ApproxVC in Figure 4 on the graph K', the integer t, and the value δ, to obtain the $(1+\delta)$-approximations $E_1, \ldots, E_m$. Finally, the algorithm outputs the list $(E_1, \ldots, E_m)$.

Analysis. By the same analysis as in the proof of Theorem 4.15 the conflict-decoder does not remove any edge between two honest servers, and since $n-2t \geq t+1$, the graph K' is a t-edge-neighborhood graph, so the weak conflict decoder runs in time $\mathrm{poly}(n)$. To see that the weak conflict-decoder satisfies the relaxed requirements, we note that

- every explanation is of size at most $(1+\delta)t$,

- validity of explanations hold as no edge between two honest servers was removed from K', and every explanation is a vertex cover of K',
- for good inputs, assume that the shares of the honest parties are consistent with some symmetric trivariate polynomial $f(x,y,z)$ of degree at most t in each variable, and observe that the set B is a vertex cover of K', so by the correctness of ApproxVC, it outputs a vertex cover E_i such that $|E_i \setminus \mathsf{B}| \leq 2\delta t$; in addition, for every explanation E the set $\mathsf{H} \setminus E$ contains at least $(n-t) - (1+\delta)t \geq t+1$ honest parties that are pairwise consistent, so they fully define the polynomial $f(x,y,z)$,
- for bad inputs, observe that for every explanation E the set $\mathsf{H} \setminus E$ contains at least $(n-t) - (1+\delta)t \geq t+1$ honest parties that are pairwise consistent, so they fully define some symmetric trivariate polynomial $f(x,y,z)$ of degree at most t in each variable.

Polynomial time VSS. In order to obtain a polynomial time VSS protocol we simply replace every execution of the conflict-decoder with an execution of the weak conflict-decoder. In more details, we make the following changes.

- In Protocol wcom, we change Step (2) in the local computation of wcom to: "Execute the weak conflict-decoder algorithm of the punctured code $\mathcal{C}_M$ on K to obtain explanations $E_1, \ldots, E_m$ of size at most $(1+\delta)(t-\ell)$". (Note that this step is anyway executed by the outer vss protocol.)
- In Algorithm Find$E$ we change Step 1 as follows: "Let $M := \mathsf{P} \setminus L$. Execute the weak conflict-decoder algorithm of the punctured code $\mathcal{C}_M$ on K to obtain the explanations $S_1, \ldots, S_m$ of size at most $(1+\delta)(t-|L|)$".
- In Algorithm Find$E$ we change Step (2,b,i) as follows: "Let $M_j := \mathsf{P} \setminus L_j$. Execute the weak conflict-decoder algorithm of the punctured code $\mathcal{C}_{M_j}$ on K_j to obtain the explanations $S_{j,1}, \ldots, S_{j,m_j}$ of size at most $(1+\delta)(t-|L_j|)$".

We note that correctness still holds. Indeed, in wcom the exact same analysis shows that the protocol is correct after the modification. For vss, it is not hard to verify that the running time of Exact$E$ is $\mathrm{poly}(n)$, and that whenever the algorithm does not return `Failure`, the algorithm returns sets that satisfy the requirements in Protocol vss. In addition, it is not hard to verify that, similarly to Claim 5.7, whenever the dealer is honest, the modified Find$E$ always return sets $(E, E_1, \ldots, E_n, I)$ that satisfy the requirements. Therefore, the same correctness analysis proves that Protocol vss remains correct after the modification. Denote the modified vss protocol by $\mathsf{vss}_{\mathrm{poly}}$, and observe that we obtain the following theorem.

Theorem 5.9. *Let κ be the security parameter, let $\epsilon > 0$ be a constant, let n be the number of parties, and let t be the number of corrupt parties, so that $n \geq (3+\epsilon)t$. Protocol $\mathsf{vss}_{\mathrm{poly}}$ is a UC-secure implementation of $\mathcal{F}_{\mathsf{vss}}$ with statistical security against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is* $\mathrm{poly}(n, \kappa, \log|\mathbb{F}|)$.

5.2 From VSS to General MPC

The work of [AKP20] presents a general transformation from an implementation of $\mathcal{F}_{\mathsf{vss}}$ to the computation of a general functionality. More concretely, for any functionality $\mathcal{F}$ that can be represented as a boolean circuit of size s and depth d, they provided a compiler that transforms any r-round protocol for $\mathcal{F}_{\mathsf{vss}}$ into an $(r+1)$-round secure realization of $\mathcal{F}$, where the compiler preserves statistical security, and has overhead $\mathrm{poly}(n, \kappa, s, 2^d)$. We therefore obtain the following theorems.[10]

Theorem 5.10. *Let κ be the security parameter, let n be the number of parties, and let t be the number of corrupt parties, so that $n \geq 3t+1$. Let $\mathcal{F}$ be an n-party functionality, represented as a boolean circuit of size s and depth d. Then there exists a UC-secure implementation of $\mathcal{F}$ with statistical security against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is $\mathrm{poly}(t^{\log t}, n, \kappa, s, 2^d)$.*

Theorem 5.11. *Let κ be the security parameter, let $\epsilon > 0$ be a constant, let n be the number of parties, and let t be the number of corrupt parties, so that $n \geq (3+\epsilon)t$. Let $\mathcal{F}$ be an n-party functionality, represented as a boolean circuit of size s and depth d. Then there exists a UC-secure implementation of $\mathcal{F}$ with statistical security against a static, active, rushing adversary corrupting up to t parties. The complexity of the protocol is $\mathrm{poly}(n, \kappa, s, 2^d)$.*

[10] Technically, the work of [AKP20] considered the functionality $\mathcal{F}_{\mathsf{vss}}$ with respect to bivariate polynomials, where the dealer inputs a symmetric bivariate polynomial $F(x, y)$ of degree at most t in each variable, and the i-th party outputs $F(x, i)$. However, it is not hard to see that the bivariate version of $\mathcal{F}_{\mathsf{vss}}$ reduces efficiently and non-interactively to the trivariate version of $\mathcal{F}_{\mathsf{vss}}$ that appears in our work in the following way: given $F(x, y)$, pick a random symmetric trivariate polynomial $G(x, y, z)$ of degree at most t in each variable, conditioned on $G(x, y, 0) = F(x, y)$, and input $G(x, y, z)$ to (the trivariate version of) $\mathcal{F}_{\mathsf{vss}}$; the i-th party, on receiving the output $G_i(x, y) = G(x, y, i)$ from $\mathcal{F}_{\mathsf{vss}}$, outputs $G_i(x, 0)$, which is equal to $F(x, i)$.

References

[AAP23] Ittai Abraham, Gilad Asharov, and Arpita Patra. Perfect asynchronous MPC with linear communication overhead. Private Communication, 2023.

[AKP20] Benny Applebaum, Eliran Kachlon, and Arpita Patra. The resiliency of MPC with low interaction: The benefit of making errors (extended abstract). In *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part II*, pages 562–594, 2020.

[ALM$^+$98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)*, 45(3):501–555, 1998.

[Alo02] Noga Alon. Testing subgraphs in large graphs. *Random Structures & Algorithms*, 21(3-4):359–370, 2002.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of np. *Journal of the ACM (JACM)*, 45(1):70–122, 1998.

[BDN16] Amey Bhangale, Irit Dinur, and Inbal Livni Navon. Cube vs. cube low degree test. *arXiv preprint arXiv:1612.07491*, 2016.

[Beh46] Felix A Behrend. On sets of integers which contain no three terms in arithmetical progression. *Proceedings of the National Academy of Sciences*, 32(12):331–332, 1946.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10, 1988.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 503–513, 1990.

[BSS04] Eli Ben-Sasson and Madhu Sudan. Robust locally testable codes and products of codes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques: 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2004, and 8th International Workshop on Randomization and Computation, RANDOM 2004, Cambridge, MA, USA, August 22-24, 2004. Proceedings*, pages 286–297. Springer, 2004.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 11–19, 1988.

[CDM00] Ronald Cramer, Ivan Damgård, and Ueli M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, pages 316–334, 2000.

[CJ03] Liming Cai and David Juedes. On the existence of subexponential parameterized algorithms. *Journal of Computer and System Sciences*, 67(4):789–807, 2003.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.

[DEL+22] Irit Dinur, Shai Evra, Ron Livne, Alexander Lubotzky, and Shahar Mozes. Locally testable codes with constant rate, distance, and locality. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 357–374, 2022.

[Din07] Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007.

[FGL+91] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost np-complete (preliminary version). In *Proceedings of the 32nd annual symposium on Foundations of computer science*, pages 2–12, 1991.

[GIKR02] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. On 2-round secure multiparty computation. In *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, pages 178–193, 2002.

[Gol10] Oded Goldreich. Short locally testable codes and proofs: A survey in two parts. *Property testing: current research and surveys*, pages 65–104, 2010.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[KKK09] Jonathan Katz, Chiu-Yuen Koo, and Ranjit Kumaresan. Improving the round complexity of VSS in point-to-point networks. *Inf. Comput.*, 207(8):889–899, 2009.

[Mei08] Or Meir. Combinatorial construction of locally testable codes. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 285–294, 2008.

[PC12] Anat Paskin-Cherniavsky. *Secure Computation with Minimal Interaction*. PhD thesis, Technion — Israel Institute of Technology, 2012.

[RAG05] Steven Roman, S Axler, and FW Gehring. *Advanced linear algebra*, volume 3. Springer, 2005.

[RB89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 73–85, 1989.

[RS78] Imre Z Ruzsa and Endre Szemerédi. Triple systems with no six points carrying three triangles. *Combinatorics (Keszthely, 1976), Coll. Math. Soc. J. Bolyai*, 18(939-945):2, 1978.

[RS92] Ronitt Rubinfeld and Madhu Sudan. Self-testing polynomial functions efficiently and over rational domains. In *SODA*, pages 23–32, 1992.

[RS97] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 475–484, 1997.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

A Appendix: Robust Conflict Decodable Codes

In this section we provide full proofs for the basic properties of robust conflict decodable codes.

Conflict checkablity plus local-to-global consistency imply robust decoding. We begin by proving Lemma 1.11.

Lemma A.1 (Lemma 1.11 restated)**.** *Let $\mathcal{C}$ be an $(n, k, d)_q$ conflict checkable code that satisfies local-to-global consistency. Then $\mathcal{C}$ is t-robust conflict decodable code for $t = \lfloor (d-1)/2 \rfloor$.*

Proof. Let $\mathcal{E}$ be a function that given a graph K on n vertices, returns (1) a single explanation $E = \varnothing$ if K contains no edges, or (2) all vertex covers of K of size at most t if K contains an edge (here, there might possibly be no explanations). Fix any word $\mathbf{x} \in [q]^n$, any set $\mathsf{B} \subseteq [n]$ of size at most t, and any graph K that is B-corrupt with respect to $\mathbf{x}$. Denote the output of $\mathcal{E}$ on K by $\mathcal{L} = (E_i)_i$.

- *(Validity of explanations)* Fix any explanation E in $\mathcal{L}$ and consider any $i, j \in \mathsf{H}$ such that $G_{i,j}(\mathbf{x}[i], \mathbf{x}[j]) = 1$. Then there is an edge (i, j) in K, and since E is a vertex cover then either $i \in E$ or $j \in E$ (or both).

- *(Good inputs)* Assume that there exists a codeword $\mathbf{c} \in \mathcal{C}$ such that $\mathbf{x}[i] = \mathbf{c}[i]$ for every $i \in \mathsf{H}$. Then for every $i, j \in \mathsf{H}$ it holds that $\mathbf{x}[i]$ is consistent with $\mathbf{x}[j]$, so there is no edge (i, j) in K. Therefore, the set B is a vertex cover of K of size at most t, so $\mathcal{L}$ is not empty. In addition, for every explanation E it holds that $|\mathsf{H} \setminus E| \geq (n-t) - t \geq n - d + 1$, and therefore $\mathbf{c}$ is the unique codeword that satisfies $\mathbf{x}[i] = \mathbf{c}[i]$ for all $i \in \mathsf{H} \setminus E$, as required.

- *(Bad inputs)* Assume that there is no codeword $\mathbf{c} \in \mathcal{C}$ such that $\mathbf{x}[i] = \mathbf{c}[i]$ for every $i \in \mathsf{H}$. If $\mathcal{L}$ is empty then we're done. Otherwise, for every explanation E it holds that $|\mathsf{H} \setminus E| \geq (n-t) - t \geq n - d + 1$, and by validity of explanations all parties in $\mathsf{H} \setminus E$ are pairwise consistent. Hence, by the local-to-global property, there exists a unique codeword $\mathbf{c}$ that satisfies $\mathbf{x}[i] = \mathbf{c}[i]$ for all $i \in \mathsf{H} \setminus E$.

This concludes the proof of the lemma. □

Robust codes are conflict checkable codes. We move on and prove that every robust conflict decodable code is a conflict checkable code.

Lemma A.2. *Let $\mathcal{C} \subseteq [q]^n$ be a t-robust conflict decodable code for $0 \leq t \leq n$. Then $\mathcal{C}$ is a conflict checkable code.*

Proof. Let $\mathbf{c} \in \mathcal{C}$ be any codeword, and consider the case where $\mathsf{B} = \varnothing$ and the i-th server receives $\mathbf{c}[i]$. Then the conflict graph K is empty, and by the guarantees for good inputs we are guaranteed that the explanation $E = \varnothing$ appears in the explanation-list that the conflict decoder generates.

Let $\mathbf{x} \in [q]^n$ and assume that $G_{i,j}(\mathbf{x}[i], \mathbf{x}[j]) = 0$ for all $1 \leq i < j \leq n$. Our goal is to prove that $\mathbf{x}$ is a codeword. Assume towards contradiction that $\mathbf{x}$ is not a codeword, and consider the case where $\mathsf{B} = \varnothing$ and the i-th server receives $\mathbf{x}[i]$. Then the conflict graph K is empty, and by the above observation, the explanation $E = \varnothing$ appears in the explanation-list that the conflict decoder generates. But from the guarantees for bad inputs there must exist a unique codeword $\mathbf{c} \in \mathcal{C}$ such that $\mathbf{x}[i] = \mathbf{c}[i]$ for all $i \in [n]$, i.e., $\mathbf{x} = \mathbf{c} \in \mathcal{C}$, in contradiction. □

Robust codes are conflict decodable codes. We continue by proving the every robust conflict decodable code is also a conflict decodable code.

Lemma A.3. *Let $\mathcal{C} \subseteq [q]^n$ be a t-robust conflict decodable code for $0 \leq t \leq n$. Then $\mathcal{C}$ is a t-conflict decodable code.*

Proof. Let $\mathcal{E}$ be the conflict-decoder of $\mathcal{C}$, and let F be the algorithm that, on a conflict graph K does as follows: F executes $\mathcal{E}$ on K to obtain the list $\mathcal{L}$, takes the first explanation E from the list and outputs the set $I = [n] \setminus E$. We continue by proving that $\mathcal{C}$ is a t-conflict decodable code with respect to F.

Let $\mathbf{w} \in [q]^n$ be any vector that is at most t-far from some codeword $\mathbf{c} \in \mathcal{C}$, let $S \subseteq [n]$ be the set of indices where $\mathbf{w}$ differs from $\mathbf{c}$, and let K be the conflict graph of $\mathbf{w}$. First, we prove that $\mathcal{E}$ on K returns a non-empty list $\mathcal{L}$. Indeed, as $\mathcal{C}$ is a robust conflict decodable code, consider the case where $\mathbf{x} = \mathbf{c}$, $\mathsf{B} = S$ and the the B-corrupt graph is K. Then, by the guarantees for good inputs, the conflict-decoder $\mathcal{E}$ on K returns a non-empty list $\mathcal{L}$, and we let E be the first explanation in $\mathcal{L}$. We are also promised that the only codeword $\mathbf{c}' \in \mathcal{C}$ that satisfies $\mathbf{c}'[i] = \mathbf{c}[i]$ for all $i \in [n] \setminus (S \cup E)$ is $\mathbf{c}' = \mathbf{c}$.

Our goal now is to prove that the codeword $\mathbf{c}$ is the unique codeword that satisfies $\mathbf{c}[i] = \mathbf{w}[i]$ for all $i \in [n] \setminus E$. As $\mathcal{C}$ is a robust conflict decodable code, consider the case where $\mathbf{x} = \mathbf{w}$, $\mathsf{B} = \varnothing$ and the the B-corrupt graph is K. Then by the guarantees for bad inputs, there exists a unique codeword $\mathbf{c}'$ that satisfies $\mathbf{c}'[i] = \mathbf{w}[i]$ for all $i \in [n] \setminus E$. Observe that $\mathbf{w}[i] = \mathbf{c}[i]$ for all $i \in [n] \setminus S$, and therefore it holds that $\mathbf{c}'[i] = \mathbf{c}[i]$ for all $i \in [n] \setminus (S \cup E)$. Hence it must hold that $\mathbf{c}' = \mathbf{c}$, and the claim follows. $\square$

As an immediate corollary, we obtain that $t \leq \lfloor (d-1)/2 \rfloor$, since this bound must hold in t-conflict decodable codes.

On the necessity of local-to-global consistency. Lemma 1.11 shows that every $(n,k,d)_q$ conflict decodable code is a t- robust conflict decodable code. The following lemma shows that local-to-global consistency is indeed necessary.

Lemma A.4. *Let $\mathcal{C}$ be an $(n,k,d)_q$ t-robust conflict decodable code, for $0 \leq t \leq \lfloor (d-1)/2 \rfloor$. Then every set $I \subseteq [n]$ of size $\ell \geq n - 2t$, and every $(x_i)_{i \in I} \in [q]^\ell$ such that x_i is consistent with x_j for all $i, j \in I$, there exists a unique codeword $\mathbf{c} \in \mathcal{C}$ that satisfies $\mathbf{c}[i] = x_i$ for all $i \in I$.*

Proof. Let $I \subseteq [n]$ be a set of size $\ell \geq n - 2t$, and let $(x_i)_{i \in I} \in [q]^\ell$ such that x_i is consistent with x_j for all $i, j \in I$. It is enough to prove that there exists a codeword $\mathbf{c} \in \mathcal{C}$ that satisfies $\mathbf{c}[i] = x_i$ for all $i \in I$, since uniqueness follows from the fact that $d \geq 2t + 1$.

Let B_1 and B_2 be a partition of $[n] \setminus I$ into two sets of size at most t. We assume without loss of generality that B_1 is not empty. First, assume that $\mathsf{B}_2 = \varnothing$. Consider the case where $\mathsf{B} = \mathsf{B}_1$, the input to the honest servers is $(x_i)_{i \in I}$ (the input to the corrupt servers is arbitrary), and let K be the B-corrupt graph with respect to $(x_i)_{i \in I}$, where every corrupt server is consistent with all other servers. Observe that K has no edges, and therefore there exists an explanation $E = \varnothing$. (Indeed, consider the case where there are no corruptions and the input to the honest servers is a valid codeword so the corresponding inconsistency graph has no edges – then the guarantees for good inputs imply that $E = \varnothing$ has to be in $\mathcal{L}$.) Hence, there exists a unique codeword $\mathbf{c} \in \mathcal{C}$ that satisfies

$\mathbf{c}[i] = \mathbf{x}[i]$ for all $i \in I$, as required. Therefore, in the rest of the proof we assume that B_2 is not empty.

Assume towards contradiction that there is no codeword $\mathbf{c} \in \mathcal{C}$ satisfying $\mathbf{c}[i] = x_i$ for all $i \in I$. Let $\mathbf{c} \in \mathcal{C}$ be any codeword, and consider the following cases.

1. *(Case 1)* Let $\mathsf{B} = \mathsf{B}_1$ be the set of corrupt servers, let $\mathbf{x}_1[i] = \mathbf{c}[i]$ for all $i \in \mathsf{B}_2$ and $\mathbf{x}_1[i] = x_i$ for all $i \in I$ be the inputs of the servers, and let K be the B-corrupt graph with respect to $\mathbf{x}_1$, where every corrupt server is consistent with all other servers. Observe that all edges in K are of the form (u, v) for $u \in \mathsf{B}_2$ and $v \in I$.

2. *(Case 2)* Let $\mathsf{B} = \mathsf{B}_2$ be the set of corrupt servers, let $\mathbf{x}_2 = \mathbf{c}$ be the inputs of the servers, and let K be the B-corrupt graph with respect to $\mathbf{x}_2$. That is, we take the same graph as in Case 1, and this is possible since all edges in K are of the form (u, v) for $u \in \mathsf{B}_2$ and $v \in I$.

Observe that there must be some explanation in Case 1. Indeed, if no explanation exists in Case 1, then, since the inconsistency graph in Case 1 is the same as the inconsistency graph in Case 2, no explanation exists in Case 2, which is a contradiction to the guarantees of good inputs.

Let E be any explanation in Case 1. Observe that there must be $i \in I$ such that $i \in E$. Indeed, if $I \cap E = \varnothing$, then, from the guarantees for bad inputs in Case 1, there exists a codeword $\mathbf{c} \in \mathcal{C}$ that satisfies $\mathbf{c}[i] = \mathbf{x}_1[i] = x_i$ for all $i \in I$, in contradiction to the assumption that such a codeword does not exist.

Finally, since the explanations in Case 1 are the same as the explanations in Case 2, we obtain that every explanation in Case 2 contains some index i that belongs to I. Therefore, in Case 2 no explanation is contained in B_2, in contradiction to the guarantees for good inputs. This concludes the proof. $\square$

A.1 Proof of Lemma 1.10

In this section we prove Lemma 1.10. We do so by proving the following claims.

Claim A.5. *Let $\mathcal{C}$ be an $(n, k, d)_q$ t-robust conflict decodable code, for $0 \le t \le \lfloor (d-1)/2 \rfloor$, that satisfies the strong guarantees for bad inputs. Then $d \ge 3t + 1$.*

Proof. Assume towards contradiction that $d \le 3t$. Let $\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C}$ be two distinct codewords of distance d, let $I \subseteq [n]$ be the set of coordinates $i \in [n]$ such that $\mathbf{c}_1[i] \ne \mathbf{c}_2[i]$, let S_1, S_2, S_3 be a partition of I into sets of size at most t (since $d \ge 2t + 1$ no set is empty), and let $S_4 := [n] \setminus I$. Consider the following cases.

1. *(Case 1)* Let $\mathsf{B} = \varnothing$ be the set of corrupt servers, and define the input $\mathbf{x}_1$ as follows: for $i \in S_1 \cup S_2 \cup S_4$ we set $\mathbf{x}_1[i] = \mathbf{c}_1[i]$, and for $i \in S_3$ we set $\mathbf{x}_1[i] = \mathbf{c}_2[i]$. Let K_1 be the corresponding inconsistency graph, and observe that all the edges are from $S_1 \cup S_2$ to S_3.

2. *(Case 2)* Let $\mathsf{B} = S_3$ be the set of corrupt servers, let $\mathbf{x}_2 = \mathbf{c}_1$ be the input of the servers. The corresponding B-corrupt graph, denoted K_2, is obtained from K_1 by removing all edges between B_3 and B_1. That is, the only edges in K_2 are from S_3 to S_2.

3. *(Case 3)* Let $\mathsf{B} = S_2$ be the set of corrupt servers, let $\mathbf{x}_3 = \mathbf{c}_2$ be the input of the servers, and set the B-corrupt graph to be K_2. (This is possible since all edges in K_2 are of the form (u, v) for $u \in S_2$ and $v \in S_3$.)

4. *(Case 4)* Let $\mathsf{B} = S_1$ be the set of corrupt servers, and define the input $\mathbf{x}_4$ as follows: for $i \in S_2$ we set $\mathbf{x}_4[i] = \mathbf{c}_1[i]$, and for $i \in S_3 \cup S_4$ we set $\mathbf{x}_4[i] = \mathbf{c}_2[i]$. Set the corresponding B-corrupt graph to be K_2. (This is possible since the conflict pattern between S_2 and S_3 in K_2 is exactly like in K_1.)

In Case 2 we obtain an explanation $E_3 \subseteq S_3$. Since the inconsistency graph in Case 2 and in Case 4 is K_2, then explanation E_3 is an explanation in Case 4 as well. Let $\mathbf{c}^*$ be the unique codeword that satisfies $\mathbf{c}^*[i] = \mathbf{x}_4[i]$ for all $i \in (S_2 \cup S_3 \cup S_4) \setminus E_3$. Observe that $\mathbf{c}^*$ agrees with $\mathbf{c}_1$ on the indices in $S_2 \cup S_4$, i.e., on at least $|S_2| + |S_4| \geq 1 + (n-d) > n-d$ indices, so necessarily $\mathbf{c}^* = \mathbf{c}_1$.

In Case 3 we obtain an explanation $E_2 \subseteq S_2$. Since the inconsistency graph in Case 3 and in Case 4 is K_2, then explanation E_2 is an explanation in Case 4 as well. Let $\mathbf{c}'$ be the unique codeword that satisfies $\mathbf{c}'[i] = \mathbf{x}_4[i]$ for all $i \in (S_2 \cup S_3 \cup S_4) \setminus E_2$. Observe that $\mathbf{c}'$ agrees with $\mathbf{c}_2$ on the indices in $S_3 \cup S_4$, i.e., on at least $|S_3| + |S_4| \geq 1 + (n-d) > n-d$ indices, so necessarily $\mathbf{c}' = \mathbf{c}_2$. However, by the strong definition we should have $\mathbf{c}' = \mathbf{c}_1$, in contradiction. This completes the proof of the claim. $\square$

Claim A.6. *Let $\mathcal{C}$ be an $(n, k, d)_q$ t-robust conflict decodable code with $d \geq 3t + 1$. Then $\mathcal{C}$ satisfies the strong guarantees for bad inputs.*

Proof. Fix any set B of size t, any bad input $\mathbf{x}$ (i.e., $\mathbf{x}$ is not a codeword), and any B-corrupt graph with respect to $\mathbf{x}$, denoted K. Let $\mathcal{L} = (E_1, \dots, E_m)$ be the corresponding list of explanations (if there are no explanations then we're done).

For every explanation E_ℓ, denote by $\mathbf{c}_\ell$ the unique codeword that satisfies $\mathbf{c}_\ell[i] = \mathbf{x}[i]$ for all $i \in \mathsf{H} \setminus E_\ell$. We prove that for every pair of explanations E_ℓ and E_r it holds that $\mathbf{c}_\ell = \mathbf{c}_r$. Let $S = \mathsf{H} \setminus (E_\ell \cup E_r)$, and observe that $|S| \geq 1$ since $n \geq d \geq 3t+1$, and $|E_\ell|, |E_r|, |\mathsf{B}| \leq t$. Observe that $\mathbf{c}_\ell$ and $\mathbf{c}_r$ agree on all indices in S. In addition, the distance of $\mathcal{C}$ when restricted to coordinates in S is at least $d - 3t \geq 1$, and since $\mathbf{c}_\ell$ and $\mathbf{c}_r$ agree on all indices in S it must hold that $\mathbf{c}_\ell = \mathbf{c}_r$, as required. This completes the proof of the claim. $\square$

B Appendix: Multilinear Forms

B.1 Proof of Lemma 4.9

In this section we prove Lemma 4.9. First, we need the following claim.

Claim B.1. *Let $m, t \geq 1$ be integers, let V be a vector space of dimension t over a field $\mathbb{F}$, let $\mathbf{v}_1, \dots, \mathbf{v}_t \in V$ be a basis of V, and let $F_1, \dots, F_t$ be symmetric $(m-1)$-forms that satisfy $F_i(\mathbf{v}_j, \cdot, \dots, \cdot) = F_j(\mathbf{v}_i, \cdot, \dots, \cdot)$ for all $i \neq j$. Then there exists a unique m-form F that satisfies*

$$F(\mathbf{v}_i, \cdot, \dots, \cdot) = F_i(\cdot, \dots, \cdot),$$

for all $i \in [t]$. In addition, F is symmetric.

Proof of Claim B.1. For vectors $\mathbf{u}_1, \dots, \mathbf{u}_m \in V$ where $\mathbf{u}_i = \sum_{j=1}^{t} \alpha_{i,j} \cdot \mathbf{v}_j$ we define

$$F(\mathbf{u}_1, \dots, \mathbf{u}_m) := \sum_{i_1, \dots, i_m \in [t]} \alpha_{1,i_1} \cdot \ldots \cdot \alpha_{m,i_m} \cdot F_{i_1}(\mathbf{v}_{i_2}, \dots, \mathbf{v}_{i_m}).$$

To see that F is indeed an m-form, note that

$$
\begin{aligned}
&F(\mathbf{u}_1, \dots, \mathbf{u}_{j-1}, \gamma\mathbf{u}_j + \delta\mathbf{u}'_j, \mathbf{u}_{j+1}, \dots, \mathbf{u}_m) \\
&\quad = \sum_{i_1,\dots,i_m \in [t]} \alpha_{1,i_1} \cdot \dots \cdot \alpha_{j-1,i_{j-1}} \cdot (\gamma\alpha_{j,i_j} + \delta\alpha'_{j,i_j}) \cdot \alpha_{j+1,i_{j+1}} \dots \cdot \alpha_{m,i_m} \cdot F_{i_1}(\mathbf{v}_{i_2}, \dots, \mathbf{v}_{i_m}) \\
&\quad = \gamma \sum_{i_1,\dots,i_m \in [t]} \alpha_{1,i_1} \cdot \dots \cdot \alpha_{j-1,i_{j-1}} \cdot \alpha_{j,i_j} \cdot \alpha_{j+1,i_{j+1}} \dots \cdot \alpha_{m,i_m} \cdot F_{i_1}(\mathbf{v}_{i_2}, \dots, \mathbf{v}_{i_m}) \\
&\qquad + \delta \sum_{i_1,\dots,i_m \in [t]} \alpha_{1,i_1} \cdot \dots \cdot \alpha_{j-1,i_{j-1}} \cdot \alpha'_{j,i_j} \cdot \alpha_{j+1,i_{j+1}} \dots \cdot \alpha_{m,i_m} \cdot F_{i_1}(\mathbf{v}_{i_2}, \dots, \mathbf{v}_{i_m}) \\
&\quad = \gamma F(\mathbf{u}_1, \dots, \mathbf{u}_{i-1}, \mathbf{u}_i, \mathbf{u}_{i+1}, \dots, \mathbf{u}_m) + \delta F(\mathbf{u}_1, \dots, \mathbf{u}_{i-1}, \mathbf{u}'_i, \mathbf{v}_{i+1}, \dots, \mathbf{u}_m),
\end{aligned}
$$

for every $j \in [m]$, every vectors $\mathbf{u}_1, \dots, \mathbf{u}_m, \mathbf{u}'_j \in V$ and every $\gamma, \delta \in \mathbb{F}$. To see that F is symmetric, note that by the symmetry of $F_1, \dots, F_t$ and since $F_i(\mathbf{v}_j, \cdot, \dots, \cdot) = F_j(\mathbf{v}_i, \cdot, \dots, \cdot)$ for all $i \neq j$, it holds that

$$F_{i_1}(\mathbf{v}_{i_2}, \dots, \mathbf{v}_{i_m}) = F_{i_{\pi(1)}}(\mathbf{v}_{i_{\pi(2)}}, \dots, \mathbf{v}_{i_{\pi(m)}}),$$

for every $i_1, \dots, i_m \in [t]$ and every permutation π of $[m]$. Therefore, for every vectors $\mathbf{u}_1, \dots, \mathbf{u}_m \in V$ it holds that

$$
\begin{aligned}
F(\mathbf{u}_{\pi(1)}, \dots, \mathbf{u}_{\pi(m)}) &= \sum_{i_1,\dots,i_m \in [t]} \alpha_{\pi(1),i_1} \cdot \dots \cdot \alpha_{\pi(m),i_m} \cdot F_{i_1}(\mathbf{v}_{i_2}, \dots, \mathbf{v}_{i_m}) \\
&= \sum_{i_1,\dots,i_m \in [t]} \alpha_{1,i_{\pi^{-1}(1)}} \cdot \dots \cdot \alpha_{m,i_{\pi^{-1}(m)}} \cdot F_{i_1}(\mathbf{v}_{i_2}, \dots, \mathbf{v}_{i_m}) \\
&= \sum_{i_1,\dots,i_m \in [t]} \alpha_{1,i_{\pi^{-1}(1)}} \cdot \dots \cdot \alpha_{m,i_{\pi^{-1}(m)}} \cdot F_{i_{\pi^{-1}(1)}}(\mathbf{v}_{i_{\pi^{-1}(2)}}, \dots, \mathbf{v}_{i_{\pi^{-1}(m)}}) \\
&= \sum_{i_1,\dots,i_m \in [t]} \alpha_{1,i_1} \cdot \dots \cdot \alpha_{m,i_m} \cdot F_{i_1}(\mathbf{v}_{i_2}, \dots, \mathbf{v}_{i_m}) \\
&= F(\mathbf{u}_1, \dots, \mathbf{u}_m).
\end{aligned}
$$

Finally, for uniqueness, let F' be any m-form that satisfies $F'(\mathbf{v}_i, \cdot, \dots, \cdot) = F_i(\cdot, \dots, \cdot)$ for all $i \in [m]$. Then

$$
\begin{aligned}
F'(\mathbf{u}_1, \dots, \mathbf{u}_m) &= F'\left(\sum_{i_1 \in [t]} \alpha_{1,i_1}\mathbf{v}_{i_1}, \dots, \sum_{i_m \in [t]} \alpha_{1,i_m}\mathbf{v}_{i_m} \right) \\
&= \sum_{i_1,\dots,i_m \in [t]} \alpha_{1,i_1} \cdot \dots \cdot \alpha_{m,i_m} \cdot F'\left(\mathbf{v}_{i_1}, \dots, \mathbf{v}_{i_m}\right) \\
&= \sum_{i_1,\dots,i_m \in [t]} \alpha_{1,i_1} \cdot \dots \cdot \alpha_{m,i_m} \cdot F_{i_1}\left(\mathbf{v}_{i_2}, \dots, \mathbf{v}_{i_m}\right) \\
&= F(\mathbf{u}_1, \dots, \mathbf{u}_m).
\end{aligned}
$$

This concludes the proof of the lemma. $\square$

We continue with the proof of Lemma 4.9. Assume without loss of generality that $\mathbf{v}_1, \dots, \mathbf{v}_t$ form a basis of V, and let F be the symmetric m-form promised in Claim B.1. It only remains to prove that $F(\mathbf{v}_i, \cdot, \ldots, \cdot) = F_i(\cdot, \ldots, \cdot)$, for all $i \in \{t+1, \ldots, \ell\}$. To see this, observe that for every $i \in \{t+1, \ldots, \ell\}$ and every $j \in [t]$ it holds that $F(\mathcal{G}_i, \mathcal{G}_j, \cdot, \ldots, \cdot) = F(\mathcal{G}_j, \mathcal{G}_i, \cdot, \ldots, \cdot) = F_j(\mathcal{G}_i, \cdot, \ldots, \cdot) = F_i(\mathcal{G}_j, \cdot, \ldots, \cdot)$, and therefore, by Claim B.1 it must hold that $F(\mathcal{G}_i, \cdot, \ldots, \cdot) = F_i(\cdot, \ldots, \cdot)$, as required. This concludes the proof of the lemma.

B.2 Proof of Lemma 4.10

We prove the claim by induction on m. The base case $m = 1$ is straightforward, and we continue with the inductive step. Assume correctness for $m - 1$, and we shall prove the claim for m. Let $\mathbf{v}_{\ell+1}, \ldots, \mathbf{v}_t$ complete the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_\ell$ to a basis of V. By the induction hypothesis, the number of ways to choose $(m-1)$-forms $F_{\ell+1}, \ldots, F_t$ that satisfy $F_i(\mathbf{v}_j, \cdot, \ldots, \cdot) = F_j(\mathbf{v}_i, \cdot, \ldots, \cdot)$ for all $i \neq j \in [t]$ is given by

$$|\mathbb{F}|^{\sum_{i=0}^{t-\ell-1} \binom{(m-1)+(t-\ell-1)-i}{m-1}} = |\mathbb{F}|^{\binom{m+(t-\ell-1)}{m}}$$

where we used the known identity $\binom{n+k}{k} = \sum_{i=0}^{n} \binom{n-i+k-1}{k-1}$. By Lemma 4.9 every choice of such $F_{\ell+1}, \ldots, F_t$ fully defines a symmetric m-form F that satisfies $F(\mathbf{v}_i, \cdot, \ldots, \cdot) = F_i(\cdot, \ldots, \cdot)$ for all $i \in [\ell]$, and in addition every symmetric F that satisfies $F(\mathbf{v}_i, \cdot, \ldots, \cdot) = F_i(\cdot, \ldots, \cdot)$ for all $i \in [\ell]$ also defines the $(m-1)$-forms $(F_i(\cdot, \ldots, \cdot) := F(\mathbf{v}_i, \cdot, \ldots, \cdot))_{i \in \{\ell+1, \ldots, t\}}$ so that $F_i(\mathbf{v}_j, \cdot, \ldots, \cdot) = F_j(\mathbf{v}_i, \cdot, \ldots, \cdot)$ for all $i \neq j \in [t]$. Therefore the number of symmetric m-forms F that satisfy $F(\mathbf{v}_i, \cdot, \ldots, \cdot) = F_i(\cdot, \ldots, \cdot)$ for all $i \in [\ell]$ is exactly $|\mathbb{F}|^{\binom{m+(t-\ell-1)}{m}}$. This concludes the proof of the lemma.

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il